

Manuscript version: Author's Accepted Manuscript

The version presented in WRAP is the author's accepted manuscript and may differ from the published version or Version of Record.

Persistent WRAP URL:

<http://wrap.warwick.ac.uk/131522>

How to cite:

Please refer to published version for the most recent bibliographic citation information. If a published version is known of, the repository item page linked to above, will contain details on accessing it.

Copyright and reuse:

The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions.

Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

Publisher's statement:

Please refer to the repository item page, publisher's statement section, for further information.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk.

Improving Trust and Reputation Assessment with Dynamic Behaviour

Caroline Player

Department of Computer Science, University of Warwick

E-mail: c.e.player@warwick.ac.uk

Nathan Griffiths

Department of Computer Science, University of Warwick

E-mail: n.e.griffiths@warwick.ac.uk

Abstract

Trust between agents in Multi-Agent Systems (MAS) is critical to encourage high levels of cooperation. Existing methods to assess trust and reputation use direct and indirect past experiences about an agent to estimate their future performance, however, these will not always be representative if agents change their behaviour over time.

Real-world distributed networks such as online market places, P2P networks, pervasive computing and the Smart Grid can be viewed as MAS. Dynamic agent behaviour in such MAS can arise from seasonal changes, cheaters, supply chain faults, network traffic and many other reasons. However, existing trust and reputation models use limited techniques, such as forgetting factors and sliding windows, to account for dynamic behaviour.

In this paper, we propose RaPTaR, a method to extend existing trust and reputation models to give agents the ability to monitor the output of interactions with a group of agents over time to identify any likely changes in behaviour and adapt accordingly. Additionally, RaPTaR can provide an *a priori* estimate of trust when there is little or no interaction data (either because an agent is new or because a detected behaviour change suggests recent past experiences are no longer representative). Our results show that RaPTaR has improved performance compared to existing trust and reputation methods when dynamic behaviour causes the ranking of the best agents to interact with to change.

1 Introduction

Multi-Agent Systems (MAS) are decentralised environments in which agents must communicate and coordinate with their neighbours to achieve tasks. Trust between agents is a vital component to ensure agents interact with competent and cooperative agents to complete tasks to a high standard (Castelfranchi and Falcone, 1998). We adopt as the definition of trust the expectation an agent has in another to achieve a task satisfactorily (Gambetta, 2000). Methods to assess trust and reputation have been developed for domains including P2P networks (Kamvar et al., 2003; Tahta et al., 2015; Xiong and Liu, 2004), pervasive computing (D'Angelo et al., 2017; Klusch and Gerber, 2002), the internet of things (Chen et al., 2016), and many more. While existing trust models are often tailored to particular contexts or require domain-dependent information, recent trust models increasingly use machine learning to generalise trust models (Liu et al., 2014; Taylor et al., 2018). However, two main challenges remain, namely, how to assign a trust value in the face of no evidence, and how to capture and detect dynamic behaviours (Traverso et al.,

2017). Some trust and reputation literature states how adapting to changes in agent behaviour is an essential requirement of a trust and reputation model (Fullam et al., 2005; Salehi-Abari and White, 2012), however existing techniques used to address this, namely sliding windows and forgetting factors, are limited in highly dynamic and distributed environments, which we discuss further in Section 2.3.

Examples of context dependent causes of behaviour change in MAS include smart grid demand shifts from seasonal changes, device faults, malicious behaviour (Hales and Edmonds, 2003; Salehi-Abari and White, 2012; Srivasta et al., 2005) and phenomena such as inflation rate fluctuations which go on to affect behaviour (Harries et al., 1998). Some of these changes can be random and unpredictable while others may be cyclical patterns. Additionally, agent behaviours may change as a result of being in a group (Griffiths, 2006; Nguyen and Bai, 2018). Group and coalition formation can improve the scalability and robustness of a network by improving task delegation amongst a smaller set of trusted agents. Traditional static coalition formation algorithms are not appropriate in MAS because of their dynamic, decentralised nature, and therefore trust and reputation has proven important in dynamic coalition formation (Klusck and Gerber, 2002). Groups in a MAS impact on trust assessment because agents rely on other members of the group to delegate subtasks to, therefore an individual agent’s reputation may be reflective of group activity. Tasks are completed to a standard dependent on the behaviour of the whole group, with individual members’ contributions unknown. Assigning reputation to agents completing subtasks is considered in group trust literature (Nguyen and Bai, 2014) and is out of the scope of our work and we assume that agents from a group have a similar behaviour. Groups can also cause dynamic behaviour as the relationships between the agents of the group can change due to changes in their motivations, a changing group population, or external factors such as a changing environment (Nguyen, 2017).

In this paper, we propose RaPTaR, a method which sits on top of existing trust and reputation methods to detect and adjust to behaviour changes to supply the underlying trust and reputation algorithm with only relevant data by deleting old instances using a modification of the Adaptive Windowing (AdWin) algorithm. The point of behaviour change is identified by monitoring the window of outcomes from a group of agents using the Kolmogorov-Smirnov (K-S) statistical test. Furthermore, transitions between behaviour changes are recorded to learn patterns in group behaviour for future exploitation. RaPTaR also provides an initial assessment of an agent belonging to a group for any trust model which uses an *a priori* trust estimate, based on both recent behaviour and any learnt behavioural patterns of the group.

RaPTaR offers several contributions to improving trust and reputation assessment in groups. First, RaPTaR allows agents to react to any statistically detected changes without a strong dependence on any prior tuned parameters. RaPTaR has only one input parameter, and we demonstrate RaPTaR is not overly sensitive to this value. This allows agents to adapt to their situation, which in MAS can vary between agents. Second, inspired by the Reactive Proactive (RePro) concept drift algorithm (Wu and Zhu, 2005), agents can learn patterns in behaviour such that any future changes can be predicted. Third, RaPTaR calculates expected behaviour considering all known behaviours weighted by the probability that they are active, which addresses a limitation of RePro which can only predict the most likely behaviour at the time. Fourth, RaPTaR learns how long agents spend acting with a particular behaviour, with the intuition that if the behaviour occurs again, it may have a similar duration, and therefore a change in behaviour can be anticipated. Finally, RaPTaR includes a probability that the agent will switch to an unknown behaviour. We compare our work to sliding windows and forgetting factors, the techniques commonly used by trust and reputation models to account for changes in behaviour. We also evaluate RaPTaR with different trust algorithms, to demonstrate its performance with continuous and discrete inputs.

Our results show that RaPTaR is significantly more robust to dynamic behaviours than existing methods. If a group of agents change their behaviour such that they are no longer the most

trustworthy group, RaPTaR adapts quickly. The extent of RaPTaR’s improvement increases as the frequency of change in the ranking of agents increases. When behaviour is static, RaPTaR will perform equally to other methods as they all converge on learning the true behaviour of an agent. However, we show that in general, RaPTaR offers statistically significant improvements over other techniques for managing agent information, for all values of α in RaPTaR demonstrating.

The remainder of this paper is organised as follows. In Section 2 we describe the related work, including existing trust and reputation models, techniques to manage interaction data in trust and reputation models, and methods for concept drift detection. The RaPTaR algorithm is described in Section 3. We present the evaluation environment in Section 4, and our results in Section 5. Finally Section 6 concludes the paper.

2 Related Work

In this section, we introduce the use of trust and reputation to assess agent behaviour, techniques for coping with dynamic behaviour, and mechanisms for detecting concept drift which inspire our approach.

2.1 Trust and Reputation

Selecting the most appropriate trust and reputation model depends on what information is available, including representations of outcomes, task decomposition, agent connectivity and social connections amongst others. Additionally, some algorithms require information about the environment either in advance, or at run-time, to accurately tune the parameters of the model. RaPTaR is a complimentary method to existing trust and reputation models which improves their estimates by managing agents’ experience data to adapt to dynamic behaviours. Therefore, we include a brief review of relevant trust and reputation literature to understand the application of RaPTaR, the models we evaluate it with, and why they are limited in coping with dynamic behaviours.

An agent assessing a potential interaction partner with a trust algorithm uses their direct past experiences with them to estimate how they will behave in the future (Keung and Griffiths, 2010). Reputation mechanisms are similar, but additionally collect, aggregate and distribute feedback about agents’ past behaviour from witnesses (Resnick et al., 2000). In the remainder of this paper we refer to trust and reputation interchangeably as we use aggregated witness reports in our trust assessments. Trust and reputation literature is extensive and addresses a variety of problems which arise in trust and reputation assessment. Biased reputation sources can be statistically altered to account for liars or perception bias using models such as FIRE, TRAVOS and BLADE (Huynh and Jennings, 2004; Regan et al., 2006; Teacy et al., 2006). Similarly, HABIT is a Bayesian inference system which addresses the need for a general behaviour representation and improves computational efficiency (Teacy et al., 2012). Stereotyping techniques improve assessments of newcomers or agents for whom there is no past experience (Burnett et al., 2010; Sensoy et al., 2016). More recently, trust assessment has been viewed as a machine-learning problem with a trust model including a learning component and a predictive component (Lu and Lu, 2017; Taylor et al., 2018).

There are no existing reputation systems that focus on statistically identifying whether witness reports are representative of the target agent’s current behaviour given that it may have changed. The main contribution of RaPTaR is to provide this feature to existing trust and reputation models, and in this paper we select Beta Reputation System (BRS) (Jøsang and Ismail, 2002), Dirichelet Reputation System (DRS) (Josang and Haller, 2007), and TRAVOS (Teacy et al., 2006) as representative trust mechanisms.

A trustor agent, tr , using BRS to assess a trustee, te , can perceive interaction outcomes as good or bad, and collects the sum of good and bad experiences with te , r_{te} and s_{te} , respectively. All the past interactions tr has had with trustees are stored in a history, \mathcal{O}_{tr} . These are the input

parameters to a beta probability density function (PDF), whose expected value is the estimate of the trustee's behaviour, $trust(te)$. An agent can aggregate witness reports from available agents to improve the accuracy of the assessment. For now, we assume that these values are relevant data about the current behaviour of te . The *belief*, b_{te} , in te is their expected behaviour based solely on their previous interactions, as calculated below:

$$b_{te} = \frac{r_{te}}{r_{te} + s_{te} + 2} \quad (1)$$

BRS weights older interactions less according to a forgetting factor, described below in Section 2.3. To account for uncertainty in this belief, an *a priori*, a , is weighted by an uncertainty factor, u_{te} :

$$u_{te} = \frac{2}{r_{te} + s_{te} + 2} \quad (2)$$

which decreases as more interaction experiences are collected, signifying a higher confidence in the belief factor. The default value for the *a priori*, a , is 0.5, which assumes that an agent is neither good nor bad, unless the default can be replaced by the output of another model:

$$a_{te} = \begin{cases} x, & \text{if a model is available} \\ 0.5 & \text{otherwise} \end{cases} \quad (3)$$

This is discussed further in Section 2.2. The overall expected behaviour according to BRS, $trust(te)$, is calculated using subjective logic, namely:

$$trust(te) = b_{te} + u_{te} \times a_{te} \quad (4)$$

The values of r_{te} and s_{te} are an aggregation of good and bad interactions with te from all available trustors known as opinion providers, $op \in \mathcal{A}_{tr}$, defined as:

$$r_{te} = \sum_{op}^{\mathcal{A}_{tr}} r_{te}^{op}, \quad s_{te} = \sum_{op}^{\mathcal{A}_{tr}} s_{te}^{op} \quad (5)$$

In BRS, the witness reports are assumed to be honest, and therefore both direct and indirect experiences are weighted equally (Burnett et al., 2010; Jøsang and Ismail, 2002). If it is not possible to assume that reputation providers will be honest or unbiased, then TRAVOS can be used. TRAVOS expands on BRS, by altering witness reports to weight opinions which have statistically proven to align with the trustor's true experiences in the past. The accuracy of a witness is calculated from tr 's perspective by comparing the beta distribution of the opinion provider's reports about any te , and the beta distribution of the actual outcomes tr ultimately received. However, this is computationally expensive because it requires collating witness reports from all trustors whenever reputation information is needed.

DRS is a generalisation of BRS, which divides interaction outcomes into k possible values. While BRS is a prominent, mathematically rigorous model, the limitation of only two possible outcomes from an interaction can be unrealistic. For example, if $k = 5$, interactions might fall into one of the following categories: 1-bad, 2-mediocre, 3-average, 4-good or 5-excellent (Josang and Haller, 2007). When tr calculates the reputation of te at time t using DRS, the set of tr 's past interactions with te , $\mathcal{O}_{tr}^{te} \subset \mathcal{O}_{tr}$, are each labelled with one of k values to become the input parameters to a multivariate Dirichlet PDF. From a Dirichlet distribution, a reputation value can be calculated using point estimate representation, which is easily computable and human understandable. Generalising storing interaction outcomes from BRS, agents have an evidence vector, $\vec{R}_{te} = (R_{te}(i) | i = 1..k)$, where $R_{te}(i)$ indicates the number of interactions from \mathcal{O}_{tr}^{te} that were rated in the i^{th} category. Each category i is given a point value, $v(i) = \frac{i-1}{k-1}$, such that the values are evenly distributed in the range $[0, 1]$. An overall reputation value is a weighted average

calculated as $reputation(te) = \sum_{i=1}^k v(i)S(i)$ given that:

$$\vec{S}_{te} : (S_{te}(i) = \frac{R_{te}(i) + Ca(i)}{C + \sum_{j=1}^k R_{te}(j)}; |i = 1, \dots, k) \quad (6)$$

where each element, $S_{te}(i)$, in the vector \vec{S}_{te} describes the multinomial probability reputation of the i^{th} element given the aggregate reports in \vec{R}_{te} , and the value of C weights the importance of the *a priori*. The literature this work is proposed in, recommend a value of $C = 2$, but optionally a higher value for C will lower the importance of the *a priori* (Josang and Haller, 2007). The *a priori*, $a(i)$, is the prior probability of the i^{th} category. Without any other information, all k intervals are given an equal probability of the *a priori*. Stereotype techniques can provide a more informative *a priori* value, $a \in [0, 1]$ for an agent, which is substituted into DRS by assigning $a(i) = 1$ to the i^{th} category that a falls into, i.e. $[a \times k]$, and all other $a(i) = 0$:

$$a(i) = \begin{cases} a(i) \in \vec{a} & \text{if } a \text{ priori model available} \\ \frac{1}{k} & \text{otherwise} \end{cases} \quad (7)$$

A disadvantage of using point estimates is that no standard deviation is calculated, and so an agent who behaves well half the time but badly otherwise, will have an equally good reputation compared to an agent who consistently performs averagely.

2.2 Groups

Trust and reputation models are data based approaches which may perform poorly when assessing a new agent for which there are no previous interactions. This is a reoccurring problem in dynamic groups where agents may have had past interactions with other agents who were previously in their neighbourhood, but none with agents who they are now connected to. One possible solution is to provide the trust algorithm with an *a priori* value for behaviour, until the agent can collect direct experiences to learn from (Burnett et al., 2010; Nguyen and Bai, 2018; Teacy et al., 2012). This may be a base value representing average performance, or an inferred value based on interactions with other agents. Grouping agents together by how they appear and behave was initially proposed in the Swift Trust algorithm (Debra et al., 1995). This method is known as stereotypes, and uses the assumption that agents who are observably similar may act similarly. For example, a decision tree can be trained on past experience data, learning correlations between agents' observable features and the trust they have in that agent (Burnett et al., 2010). The *a priori* is given less precedence as direct experiences become available. HABIT uses the average value of the first interactions with other agents in the group the trustee is associated with. This represents the notion that if previously interacting with an agent in that group for the first time was typically bad, the trustee might also be bad. StereoTrust assigns agents an *a priori* assessment of their behaviour based on the trust they have in other agents who are observably similar, weighted by the extent of that similarity (Liu et al., 2009). If agents of a group can be assumed to have the same static behaviour, we extend this to assume their behaviour at any time point is similar, given that it can change.

2.3 Dynamic Behaviour

Dynamic behaviour is a challenge to trust assessment because behaviour can change at a variety of speeds and times for the agents in a MAS, rendering traditional methods of forgetting old data at a constant rate ineffective. Despite this limitation, prominent trust and reputation models still rely on techniques such as sliding windows and forgetting factors (Huynh and Jennings, 2004; Jøsang and Ismail, 2002; Liang and Shi, 2005; Regan et al., 2006; Teacy et al., 2006). While other literature mentions the importance of coping with dynamic behaviour, it remains an open challenge (Anders et al., 2016).

A sliding window of size n retains the most recent n experiences. The intuition is that the most recent n instances capture an agent’s current behaviour, and older interaction records which are no longer relevant are deleted. For a new instance, o^t , at time t and a window, W , of size n the window is updated as follows.

$$W \leftarrow (W/W[n]) + o^t \quad (8)$$

A forgetting factor, also referred to as a longevity factor, forgets at a constant rate with all instances being retained and more recent interactions weighted higher. An interaction from time t at the current time, t' , which can be used in trust assessment is weighted as:

$$o^{t,t'} = \lambda^m \times o^t \quad (9)$$

where m is the elapsed time since o^t was recorded i.e. $m = t' - t$. Forgetting factors can be implemented recursively by storing one value which is updated over time instead of storing all values, reweighting, and collating them at every time point. This improves on both time and space efficiency as older interactions do not need to be searched or saved. However, not saving old data loses potentially useful information, especially about behaviour change over time.

Unfortunately, several issues impact on the generality of using sliding windows or forgetting factors to aggregate information. First, choosing the window size or the rate of forgetting is problematic. If an agent forgets old instances too quickly they will lose relevant data that could help them make more accurate calculations. Conversely, if too much information is retained then agents will make assessments based on data that no longer represents current behaviours. The optimal values will depend on the application. Second, the optimal values for window sizes or forgetting factors can vary over time. For example, behaviours may change frequently for a period and then remain static, and this would require using very recent data initially and then a larger window size. Third, agents may not change their behaviour at the same rates, and so a global window size or forgetting factor will not suit all agents. Finally, sliding windows and forgetting factors are only effective in coping with gradual change rather than sudden changes, which may occur at different times for different agents.

Existing MAS literature outside of the scope of trust and reputation has looked at identifying the best behaviour policy to use from a set of policies, by identifying other agents’ behaviours and responding to that. Part of this task, involves identifying when other agents have changed their own policy (Hernandez-Leal et al., 2016). One reason we do not compare against policy change literature is in our context, agent behaviours are not a series of static policies which are swapped between, but rather are continuous values which can change gradually or suddenly. Agents need to learn behaviours through outcome rewards and cannot identify discrete action choices of other agents immediately. Identifying agent behaviour changes through analysing a continuous variable is a different problem to policy detection, and this depends on the information available in the environment (Hernandez-Leal et al., 2017). Finally, as discussed with all existing techniques on estimating agent behaviour, work on reacting to policy changes also does not discuss the probability of an upcoming change.

In this paper, we propose a method that uses *concept drift* techniques to identify changes in behaviour. Machine learning models find correlations between a feature set, X , and a class value, Y , typically assuming that the joint probability of the features and the target remains static over time, i.e. $P_t(XY) = P_{t'}(XY)$ for all times t and t' (Webb et al., 2016). Concept drift techniques aim to detect when such relationships change over time, meaning $P_t(Y|X) \neq P_{t'}(Y|X)$ (Tsymbol, 2004). In stereotyping, agents are grouped based on their observable features, and the correlation between those features and agent behaviour is learned. In this paper, we address the problems which arise from agent behaviour changing over time meaning that the correlation between features and behaviour may change.

A concept is a learned correlation between the features and the target, or class value, which in our context is the trust value. If the correlation between a set of feature values and a class value instantly changes to a new class value then *sudden* drift has occurred. A slow progression of

one concept becoming less prevalent while another becomes more prevalent is known as gradual drift. Gradual drift is harder to detect, especially if the differences between the two concepts are not empirically large but are important (Hoens et al., 2012). Sliding windows and forgetting factors will have an optimal size or value to handle only one speed of change, however agents may change behaviour at different times and rates and a single window size or forgetting factor may be insufficient. The method proposed in this paper handles agent behaviours changing at different speeds or at different times without requiring parameter tuning.

In our approach, we draw upon concept drift techniques which can identify the point of change in the data, allowing the agent to update its interaction history. The Adaptive Windowing (AdWin) model uses a variable window size which expands with new, incoming data if that data is statistically assessed to be drawn from the same distribution as the rest of the data in the window (Bifet and Gavaldà, 2007). If there is a split of the window such that it is believed the two subsets come from different distributions, then change has been detected and data from the oldest subset is forgotten, which is synonymous with shrinking the window. The method we propose in this paper uses a similar adaptive window as part of its reactive component. Once AdWin deems a concept to have ended, the data from that concept is forgotten forever. RePro retains such information with the aim of improving predictions in an environment where concepts might reoccur (Wu and Zhu, 2005). Once a concept has changed, RePro remembers the concept and records the transition to it from the previous concept. If the concept reoccurs in future, it can be pro-actively predicted given the concept which came before it, without being relearned from scratch, therefore saving time and improving predictive accuracy. RePro was designed for categorical predictions, however we present a simple adaptation for continuous predictions. RePro can also only predict one concept at a time, as the proactive mode selects the concept which most frequently follows the current concept and there is no consideration of unknown concepts. The proactive mode requires a trigger, such as detecting a small amount of error, to determine whether a different concept has taken over. Work on detecting concept drift in the error of agent behaviour estimates exists (Hernandez-Leal et al., 2017), however, in the context of trust and reputation, estimates of behaviour can be highly inaccurate but not affect the ability to choose good partners. We find it more prudent to analyse interaction outcomes for concept drift, which have not been biased by agents' potentially inaccurate trust estimates, which allows RaPTaR to detect changes effectively regardless of which trust algorithm is being used.

3 Reacting to Dynamic Agent Behaviours and Predicting Trust and Reputation

In this section, we propose a method, RaPTaR (*Reacting and Predicting in Trust and Reputation*), to detect and adapt to changes in agent behaviour such that trust assessment is as relevant as possible to an agent's current behaviour. By improving the accuracy of trust assessment, and identifying the current best partner to interact with given that agents' behaviours can change, we aim to improve the average utility agents receive. The following assumptions are made regarding the environment in which agents are situated. First, we assume that agent behaviour can change at any rate or time. Second, we assume that agents may interact with a subset of the population, as determined by an underlying network topology. An agent's connectivity limits its ability to gather reputation information, and its choice of available partners. Third, we assume that agents belong to a group, G , such that members of the group have the same behaviour. If some factor causes an agent to change its behaviour then we assume the behaviour changes for all members of the group. Fourth, we assume that agents provide accurate reputation information, because they are honest and unbiased. In principle, because RaPTaR is used alongside an underlying trust assessment mechanism, this assumption is mitigated by trust techniques that cope with dishonest agents. In this paper, our focus is on the mechanism for selecting appropriate information on which to assess reputation, rather than addressing dishonesty.

In the remainder of this section, we describe our method which improves trust assessment in two ways. First, statistically detecting changes in recent outcomes from interactions, and updating the data used in trust assessment accordingly, enables agents to assess trust appropriately for others’ behaviour. Second, RaPTaR produces an estimate of an agent’s behaviour based on its group by exploiting any learned patterns.

An agent uses RaPTaR to monitor the outcomes of a subset of agents, $G \subset \mathcal{A}$, to account for how a group may change its behaviour at different times and speeds compared to other groups. Each identifiable group in the population is monitored separately. Groups may represent some known coalitions but, if groups are not explicitly identifiable, it may be possible to group agents using techniques such as stereotypes (Player and Griffiths, 2018). Our previous work showed how we can react to dynamic behaviours of stereotypes. In this paper, we build on this idea, to also consider how we can predict behaviour changes. We found in our previous work that the level of noise prevented accurately identifying change, therefore, we aim to reduce the noise to study more complex agent behaviours. Therefore, in this paper, we assume that groups are identifiable. RaPTaR aims to make use of past learned behaviour information after the agent has changed behaviour, which is not something considered by previous methods. An agent stores the outcome of its interactions in a history, \mathcal{O} , which is divided into subsets representing the outcomes of interactions with agents from each group they have encountered, $\mathcal{O}_G \subset \mathcal{O}$. RaPTaR can then monitor \mathcal{O}_G for behavioural changes within each group G . In presenting RaPTaR, we use the notation in Table 1.

RaPTaR has a learning component and a predictive component. When the outcome of an interaction is saved, it will either be assumed to come from the same distribution as other recent outcomes, or from a different distribution if a change in behaviour is detected. For the learning component we use a modified version of the AdWin algorithm to detect change and remove irrelevant data. If a change is detected, the behaviours on either side of the change are learned. Agents record how long a behaviour was believed to be active for and which behaviour succeeded it, to learn behavioural patterns and improve predictions of future behaviour changes. Some trust and reputation algorithms will accept an *a priori* as input to improve trust estimates when there is little experience data. Similar to stereotype models, the predictive component of RaPTaR produces an *a priori* estimate based on experiences from members of the group who are assumed to behave similarly. RaPTaR uses the adaptive window of recent interactions to estimate the current behaviour, how long it has been active for, and possible successor behaviours to assess an overall expected utility. We discuss the learning and predictive components in Sections 3.1 and 3.2 respectively.

3.1 Learning Component

The learning component comprises two parts: the first detects changes in agent behaviour using interaction outcomes, and the second learns patterns in changes of behaviour.

RaPTaR maintains a window, W , of outcomes from interacting with the agents of a group, G , such that after an interaction the outcome, o^t , is appended to the end of W . For every split of W into W_0 and W_1 , we perform the K-S test to determine whether the null hypothesis, “the data in the two sets is drawn from the same distribution” can be rejected, i.e. there has been a change in the agents’ behaviour. The K-S test is appropriate because the data is continuous (Press et al., 2007). There is only one parameter in the K-S test, $\alpha \in [0, 1]$, where the confidence in the decision of accepting or rejecting the null hypothesis is $1 - \alpha$. Therefore, for a high level of confidence in this decision, α should be a low value. We show in Section 5, that in our results the efficacy of RaPTaR does not fluctuate with differing values for α .

In the original AdWin algorithm, W is iteratively split into all combinations of W_0 and W_1 starting from isolating the oldest instance and incrementally moving the oldest element of W_1 into W_0 , until either change is detected or all the instances in W are assumed to be from the same distribution. We identified that there is a performance difference in the AdWin algorithm

Table 1 RaPTaR notation

Notation	Description
$G \subset \mathcal{A}$	An identifiable group of agents assumed to behave similarly, and therefore monitored using RaPTaR.
$o_t \in [0, 1]$	The outcome of an interaction which occurs between a trustor, tr , and trustee, te , at time t .
W	An adaptive window of recent interaction outcomes that a trustor has had with a group of trustees.
$t_{W[x]}$	The time that the interaction indexed at x in window W occurred.
α	Input parameter to set the sensitivity of the K-S test. The confidence in the decision made by the K-S test is $1 - \alpha$.
$c \in \vec{\mathcal{C}}$	A concept in the list of known concepts for G which have been learnt from interactions with members of G . In our context, the value of a concept is an estimated behaviour value.
$c_c \in \vec{\mathcal{C}}$	The currently active concept which estimates the values currently in W . This will either be a concept from the history if such a concept exists that predicts W well, or a new concept otherwise.
cet	The conceptual equivalence threshold, used to determine if the values in W are similar enough to an already known concept in the concept history.
TM	A transition matrix to maintain statistics about behaviour changes, recorded as concept transitions, which are used for future predictions of agent behaviour.
$[c_x, c_y] \in \vec{\mathcal{C}}$	Variables to keep track of the two concepts preceding c_c in order to record transition $c_x \rightarrow c_y$ in TM .
s	The stable learning size, which is the minimum length of time a concept should be active for to consider it a well learned and established concept.
$L[c_x, c_y]$	A list containing the duration of time it was detected that agents spent with learned behaviour c_x before transitioning to their next behaviour, learned as c_y . Such a list exists in every cell of the TM, for every possible transition of the known concepts.
tl_{c_x}	The length of time c_x is believed to have been active before transitioning to c_y , a value stored in the list $L[c_x, c_y]$.

if W is split into W_0 and W_1 in reverse order, such that the most recent instance is isolated first into W_1 with all other instances in W_0 , then iteratively moving the last instance of W_0 into W_1 . To keep the retained data as relevant as possible, conducting the split starting at the most recent instance is more effective. This is because if there is a gradual change in the distribution over data, which does not have one clearly defined time point of change, the null hypothesis could be rejected at any of several consecutive split points. By identifying the most recent split point, the most relevant data is retained in W_1 . Alternatively, starting the tests from older instances will result in the change being detected closer to the start of the window and thus retaining more, possibly irrelevant, data in W_1 . Algorithm 1 describes how RaPTaR learns behavioural patterns, splitting W into W_0 and W_1 by starting to check for behaviour change at the end and then working backwards.

If there exists a split of W into W_0 and W_1 such that a change in behaviour is detected, RaPTaR stores information about the change for future predictions about behaviour. When a change is detected, the instances in W_0 can be learnt as a concept. The currently active concept, c_c , which best estimates W_1 , is still active and may still change, so we cannot record a transition to it yet. Therefore, transitions are recorded between the two concepts which were active prior to

c_c . Concept c_y represents the behaviour in W_0 , and c_x was the concept learned before c_y , learned from data which has since been removed from W . Concept c_x is initialised to the *unknown concept*, and then updated at the end of this process as $c_x \leftarrow c_y$, ready to record the next transition. The form of the transition matrix is depicted in Table 2. The first column of TM , index 0, is for the *unknown* concept, to show how frequently other concepts are followed by a new concept. When a change in behaviour is detected, as described above, the concept c_y which describes the behaviour in W_0 needs to be identified as either a concept seen before, or be learnt as a new concept. The value of all the known concepts in \vec{C} , is compared to the mean of the instances in W_0 , μ_{W_0} , and if there exists a concept $c \in \vec{C}$ such that $|\mu_{W_0} - \mu_c| < cet$, where cet is the *conceptual equivalence threshold*, then $c_y \leftarrow c$. Storing mean values is a simple approach, and future work might consider storing more complex agent policies. If no concept is equivalent, a new concept is learnt with the value μ_{W_0} . If the length of W_0 is less than a stable learning size, s , then it is not considered substantial to record a transition for. The previous concept, c_x , is remembered even though the data for it has since been deleted, and we can now record the transition between c_x and the newly learned c_y . The list, $L_{[c_x, c_y]}$, records the length of time spent at c_x , tl_{c_x} , before moving to c_y , for every occurrence of the transition. The time at c_x is initialised to 0 for the first recording. The reactive and learning component is summarised in Algorithm 1.

Table 2 Transition Matrix

Previous Concept, c_x	Successor Concept, c_y				
	unknown	1	2	3	...
1		$L_{[1,1]}$...
2					...
3					...
...					

3.2 Predictive Component

For any trust and reputation model which accepts an *a priori* trust estimate, RaPTaR calculates an expected utility based on the value of each concept, $\mu_c \in \vec{C}$, in the concept history for a group, and the probability it may currently be active, $p(c)$. The expected utility draws upon the information recorded in TM by the learning component and is calculated as:

$$EU = \forall_{c \in \vec{C}} \mu_c \times p(c) \quad (10)$$

First, tr identifies if any of the known concepts, $c \in \vec{C}$, could be the currently active concept, c_c , by evaluating the difference between the concept values, μ_c , and the mean of the values currently in W , μ_W , if $|\mu_c - \mu_W| < cet$. The trustor assesses the length of time this concept has likely been active, t_{c_c} , as the current time, t , minus the time the first outcome in W was recorded:

$$t_{c_c} = t - t_{W[0]}$$

The first known occurrence of the currently active concept should be in the first element of the adaptive window, $W[0]$, as the reactive component statistically readjusted the window to contain data believed to be generated by the same underlying distribution.

The next step is to calculate the probability, $p(c)$, that each concept $c \in \vec{C}$ will succeed c_c given that c_c is assumed to have been active for t_{c_c} . A PDF is built with a Gaussian Kernel Density Estimator using the lengths of time spent at c_c in the past before it was succeeded by c . These times were recorded in TM, illustrated in Table 2, $\forall tl_{c_c, c} \in L_{[c_c, c]}$. A PDF interprets the probability of the length of the concept, which is advantageous because the reactive component may not identify the time of change perfectly, but the PDF will combine all the times to give

Algorithm 1 Learning Concepts by Updating TM

```

function UPDATE( $o_{te}^t$ )
   $W \leftarrow W + o_{te}^t$ 
  for ( $i = |W| - s; i > s; i - 1$ ) do
     $W_0 \leftarrow W[0, i]$ 
     $W_1 \leftarrow W[i, |W|]$ 
    if  $KS - Test(W_0, W_1) < \alpha$  then
       $c_y \leftarrow LearnConcept(W_0)$ 
       $TM[c_x, c_y, L] \leftarrow TM[c_x, c_y, L + \{tl_{c_x}\}]$ 
       $c_x \leftarrow c_y$ 
       $tl_{c_x} \leftarrow t_{W_0[|W_0|]} - t_{W_0[0]}$ 
       $W \leftarrow W_1$ 
      return
    end if
  end for
end function

function LEARNCONCEPT( $W_0$ )
   $c_y \leftarrow null$ 
   $\mu_{W_0} \leftarrow avg(W_0)$ 
  for  $c \in \vec{C}$  do
    if  $|\mu_c - \mu_{W_0}| < cet$  then
       $c_y \leftarrow c$ 
    end if
  end for
  if  $c_y == null$  then
     $TM[c_x, 0, L] \leftarrow TM[c_x, 0, L + \{tl_{c_x}\}]$ 
     $c_y \leftarrow |TM| + 1$ 
     $\mu_{c_y} \leftarrow \mu_{W_0}$ 
    expand  $TM$  to include new concept
  end if
  return  $c_y$ 
end function

```

an estimate. As $|L_{[c_c, c]}|$ increases, the PDF will get more accurate at pinpointing the time. The probability, $p(c)$, that c succeeds c_c after t_{c_c} time at c_c is:

$$p(c) = PDF(t_{c_c}) \times |L_{[c_c, c]}|$$

where $|L_{[c_c, c]}|$ is the length of the list, and therefore the number of times c has succeeded c_c . Multiplying by the frequency of transitions turns the probability into a frequency distribution estimate which will give precedence to the concepts which more frequently succeed c_c when the probabilities are normalised.

To normalise the probabilities of each possible next concept (including the possibility that c_c is followed by c_c at this time), the probabilities must sum to one, $\sum_{c \in \vec{C}} p(c) = 1$. Therefore, $p(c)$ is given as:

$$p(c) = \frac{p(c)}{\sum_{c_n \in \vec{C}} p(c_n)} \quad (11)$$

If no known concept is active, the currently active concept is assumed to be new and it takes on the value μ_W , and probability 1. No other concept has a probability of being active as we have no information about what or when another behaviour might follow the currently active concept.

3.3 Integrating RaPTaR with trust and reputation models

RaPTaR has two points of integration with trust and reputation algorithms. First, RaPTaR detects changes in the behaviour of agents in a group, G , and deletes any records believed to be irrelevant from \mathcal{O} . The effect is that the trust and reputation model draws upon records in \mathcal{O} which are believed to be representative of current behaviours. This contrasts with use of a fixed size sliding window that can only delete the oldest interaction record to accommodate space for the newest interaction record, regardless of the relevance of either the newer or older record. Second, RaPTaR outputs an expected utility which can be used as an *a priori* estimate in trust and reputation models which incorporate such a value when there are no, or few, direct experiences with an agent on which to otherwise base a decision.

4 Experimental Methodology

In this section, we describe the evaluation environment used in this paper. RaPTaR is an extension to trust assessment requiring minimal parameter tuning, and therefore RaPTaR is applicable in multiple domains. We describe general environmental parameters which can be altered to model a variety of application types.

An interaction between two agents can represent an exchange of goods or services between producer and consumer. A trustor agent, tr , makes a trust assessment about a trustee agent, te , and if they are the most trustworthy available agent, they have an interaction. Agents interact over time steps and gain experiences with other agents which they use to assess their future performance. Agents behave as both trustors and trustees. In a round, a trustor chooses a partner to execute their task, but they can also be selected as a trustee by other agents. In our evaluation, an agent can act as a trustee in up to 10 requests per time step. This mimics evaluations from existing literature where there are more trustees than trustors, as well as enabling agents to multi-task (Nguyen and Bai, 2018). When there are more possible partners to choose from, trustors have a more difficult task of identifying the best partner, therefore, the results showcase which partner selection technique is more successful.

An agent is described by the tuple $\langle ID, G, b(t) \rangle$ where ID is the agent’s identifier, G is the agent’s groups and $b(t) \in [0, 1]$ describes agents’ behaviour which is dependent on time t and their group. The behaviour at any time point is described below. An agent maintains a set of historical records of their past interactions, \mathcal{O} , where each tuple represents one interaction in the form: $\langle j, o_t, t \rangle$ where j is the ID of their interaction partner and o_t is the value of the interaction at time t .

4.1 Dynamic Behaviours

The trustor, tr , receives an outcome as a result of interacting with a trustee te , $o_t \in [0, 1]$, dependent on the behaviour of te at time t . This value can be seen as the extent to which, or the probability that, te cooperates or defects. Static agent behaviour, b , is typically defined within a range of values where the bounds are application dependent. Trust and reputation algorithms then normalise these bounds such that $b \in [0, 1]$, where 0 indicates defective behaviour while a value tending towards 1 indicates a cooperative agent. We extend this to define dynamic behaviours, $b(t) \in [0, 1]$, as a function dependent on time. These functions are randomly generated each experiment, so that the length of time concepts are active for, and the speed of transitions vary. our evaluation then shows how each method copes given there is no advance information about agent behaviours. The outcome of an interaction is then drawn from a Gaussian distribution with mean $b(t)$ and standard deviation 0.2, to represent that an agent will not always perform exactly

the same, or that agents sharing the behaviour from that group behave uniformly. Examples of dynamic behaviours are given in Figure 1. These functions could be repetitive patterns over time as seen in Figure 1(a), or be random dynamic behaviour as depicted in Figure 1(b). An agent’s behaviour is defined by the group they belong to.

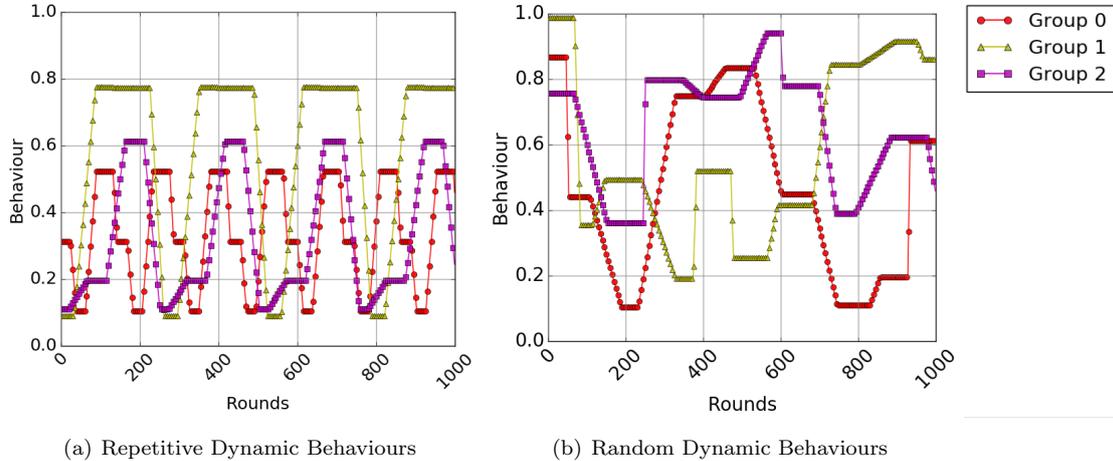


Figure 1 Example of groups’ dynamic behaviours.

4.2 Groups

In the model assumptions stated in Section 3, agents are identifiable by group, and their behaviour is dictated by their group. Figure 1 illustrates how a groups’ behaviour can change over time. Groups may be explicit, as we have assumed in this paper, but they may represent profiles of entities in the application. For example, Smart Grid users might be classified into groups including a single person living alone, or large houses with multiple occupancy, or industrial offices. As smart meters become more advanced and prevalent in the electricity grid, they can provide more information to improve these classifications using stereotype techniques when group identities are not explicit. For the evaluation in this paper we use 3 groups, and agents do not change between groups. Agents are randomly assigned to a group at the beginning of their lifetime.

4.3 Network Topology

To demonstrate that RaPTaR is appropriate in many domains we consider agents being connected in different network structures (Anderson et al., 2013). The network topology affects which agents are connected to which other agents, and the consequences to trust and reputation models include limiting available interaction partners and witnesses. Our evaluation considers three common structures. The Kleinberg small-world network where agents are grouped in hubs with few links connecting each hub, the Barabási-Albert scale-free network which uses the preferential attachment property, often used to represent social networks, where agents have a higher probability of being attached to an agent who already has a higher degree, and a fully connected network. When we use the small-world or scale-free networks in our evaluation they are populated with 100, 200 or 500 agents, but the fully connected network is only populated with 20 or 40 agents as the increased number of edges means it is significantly less scalable and simulations become too long to run. We include results with a fully connected network because this represent a portion of the network or a group rather than the whole population. Additionally, a fully connected network represents a scenario where agents have sparse data because of the high degree but many possible interaction partners. Small-world networks in the experiments are generated with a clustering coefficient of 0.5.

4.4 RaPTaR variables

The conceptual equivalence threshold, cet , is set to 0.1 in our evaluation. Trust models can have low accuracy and so a value of cet as high as 0.1 helps to group together interactions with trustees whose outcomes were generated by the same behaviour but have a margin of discrepancy between their assessment. Additionally, this is low enough that any true behaviours which are different but are assessed to be the same concept are not too different. Previous experiments show that perfect accuracy for estimating agent behaviour is not important but that the ranking of agents needs to be correct (Player and Griffiths, 2017). If $cet = 0.1$ this bins trust into 10 categories (assuming that trust is in the range $[0, 1]$), providing enough granularity to rank behaviour from good to bad. The stable learning size, s , is set as low as 3 to encourage learning brief periods of behaviour which occur during transitions between longer periods of static behaviour. We do not consider this an input parameter to RaPTaR, because we recommend the value to be small in a dynamic noisy agent-based environment.

5 Results and Evaluation

In this section, we present experimental results of RaPTaR applied to three trust and reputation algorithms: BRS, DRS and TRAVOS. The results demonstrate that RaPTaR is an effective alternative to sliding windows and forgetting factors when the dynamic nature of the environment is unknown. All results are averaged over 100 runs and are statistically significant with a p -value less than 0.001 using a paired t-test. The only exceptions to this are in Figures 2 and 3(a) where the results demonstrate how RaPTaR works in a single experiment. RaPTaR aims to improve an agent’s ability to assess another agent’s behaviour, and thereby select better partners for interactions. Differences in results obtained from using RaPTaR with different α values are not statistically significant, demonstrating there is little dependence on this parameter. To illustrate this, the results show the average utility agents receive for interactions. For both trust models, BRS and DRS, we have put in bold the evaluation results which are best. Table 3 shows the average utility, or outcome, an agent receives across all rounds and runs. RaPTaR outperforms all sliding windows with various window sizes (ws) and forgetting factor values (λ), where the improvement varies between 2-5%. This average improvement demonstrates RaPTaR is robust and consistent, but RaPTaR provides larger improvements in times of dynamic behaviour and performs equally well as existing techniques during periods of static behaviour, thus reducing the average improvement. The extent of the improvement can vary depending on the network topology, which is discussed below. Agents dynamic behaviour is patterned in all evaluations unless otherwise stated, where we show that RaPTaR is effective for random dynamic behaviour as well.

To visualise why and how RaPTaR performs well we analyse a single run. In this example, we use a fully connected network to emphasise how agents behave when they have a lot of choice for partners and sparse data, which demonstrates the efficacy of the behaviour assessment model. We compare how BRS performs with RaPTaR compared to a sliding window. Figure 2 shows trustors’ average trust estimate of agents from each group at every time period. We can see that first, RaPTaR adjusts to changes as they occur, and second, RaPTaR makes more accurate assessments of all the groups, not just the best one. Figure 3(a) depicts the average outcome that agents receive in this experiment. At times 500-550 and 850-900 there are substantial improvements from using RaPTaR. This is not a constant improvement across all rounds because during static periods of behaviour, the sliding window eventually tends towards accuracy. However, as soon as the dynamic behaviour of the groups causes the best group to change, RaPTaR’s ability to adapt helps agents select a better partner sooner. This analysis can be verified by observing the accuracy of the trust estimates at those times in Figure 2. Additionally, RaPTaR can predict cyclical behaviour before the change has occurred, as well as react, while existing methods take a period of time to relearn behaviours even though they may have been seen before.

Table 3 Average utility per agents per time step in network topologies using different trust models with either a fixed window size (ws), forgetting factor (λ) or RaPTaR, averaged over 1000 rounds and 100 runs.

Network Type		Fully Connected		Small-World			Scale-Free			
$ \mathcal{A} $		20	40	100	196	484	100	200	500	
BRS	ws	50	0.564	0.564	0.550	0.553	0.553	0.541	0.540	0.539
		100	0.552	0.553	0.543	0.543	0.543	0.534	0.533	0.532
	λ	0.9	0.578	0.580	0.558	0.565	0.565	0.552	0.550	0.549
		0.95	0.573	0.574	0.560	0.561	0.561	0.549	0.547	0.545
	RaPTaR α	0.99	0.564	0.566	0.553	0.553	0.553	0.542	0.540	0.539
		0.05	0.590	0.591	0.573	0.573	0.573	0.556	0.554	0.553
		0.2	0.593	0.595	0.575	0.575	0.575	0.558	0.556	0.554
		0.3	0.596	0.596	0.576	0.577	0.577	0.559	0.557	0.555
		0.4	0.597	0.598	0.578	0.578	0.578	0.559	0.557	0.555
		0.5	0.599	0.599	0.579	0.579	0.579	0.560	0.558	0.556
		0.6	0.599	0.599	0.579	0.580	0.579	0.560	0.558	0.556
		0.7	0.599	0.600	0.579	0.580	0.579	0.560	0.559	0.556
0.8		0.599	0.600	0.580	0.580	0.580	0.561	0.559	0.557	
DRS		ws	50	0.564	0.566	0.550	0.554	0.554	0.542	0.540
	100		0.552	0.553	0.542	0.544	0.544	0.535	0.534	0.532
	λ	0.9	0.557	0.558	0.549	0.546	0.546	0.539	0.537	0.536
		0.95	0.554	0.556	0.544	0.544	0.545	0.537	0.536	0.534
	RaPTaR α	0.99	0.559	0.561	0.549	0.549	0.549	0.539	0.538	0.537
		0.05	0.596	0.597	0.578	0.578	0.579	0.560	0.558	0.556
		0.2	0.602	0.603	0.582	0.582	0.582	0.563	0.561	0.559
		0.3	0.605	0.606	0.584	0.585	0.584	0.564	0.562	0.560
		0.4	0.605	0.606	0.584	0.584	0.584	0.564	0.562	0.560
		0.5	0.606	0.607	0.584	0.585	0.585	0.564	0.562	0.560
		0.6	0.606	0.607	0.584	0.585	0.585	0.564	0.562	0.560
		0.7	0.607	0.607	0.585	0.585	0.585	0.564	0.562	0.560
0.8	0.607	0.607	0.585	0.586	0.585	0.564	0.562	0.560		

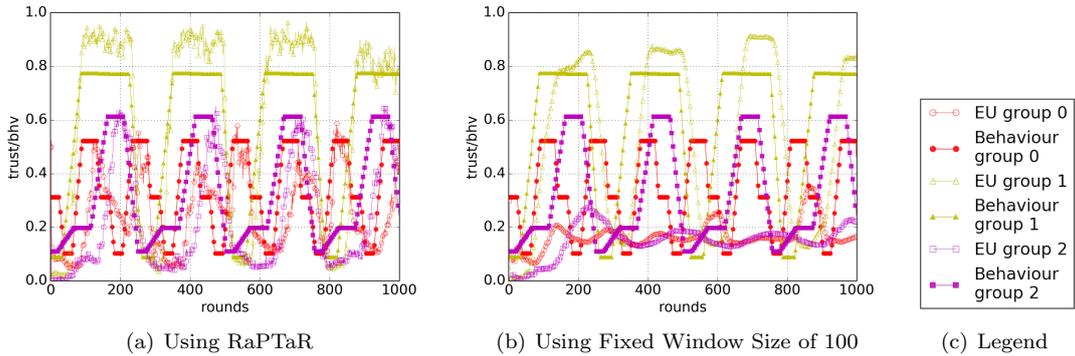


Figure 2 RaPTaR more quickly adapts its trust estimates (EU) to reflect the true behaviour compared to other methods. Agents use BRS in a fully connected network with an exploration rate of 0.2.

To test if these improvements occur for dynamic behaviours which change at different times and speeds across multiple runs, Figure 3(b) compares the average utility over 100 runs using the

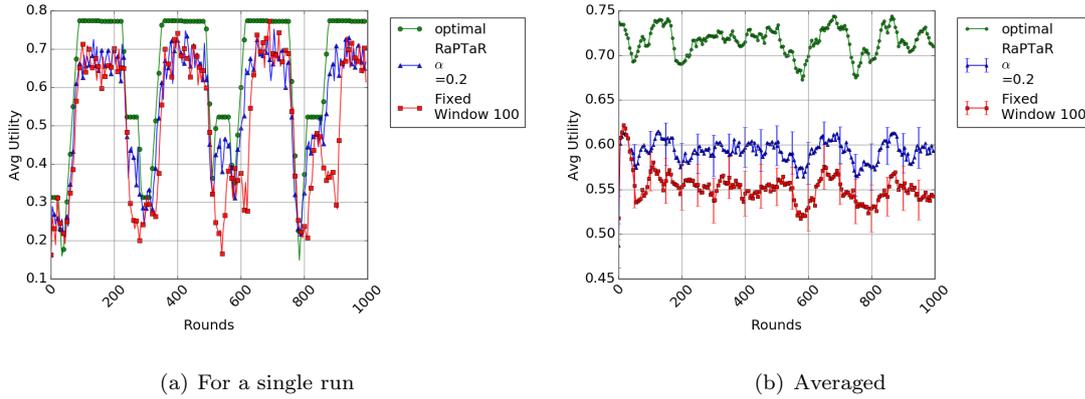


Figure 3 Quick adaptations to behaviour change lead to short term large utility benefits which are substantial enough to be significant improvements to average utility over 100 runs.

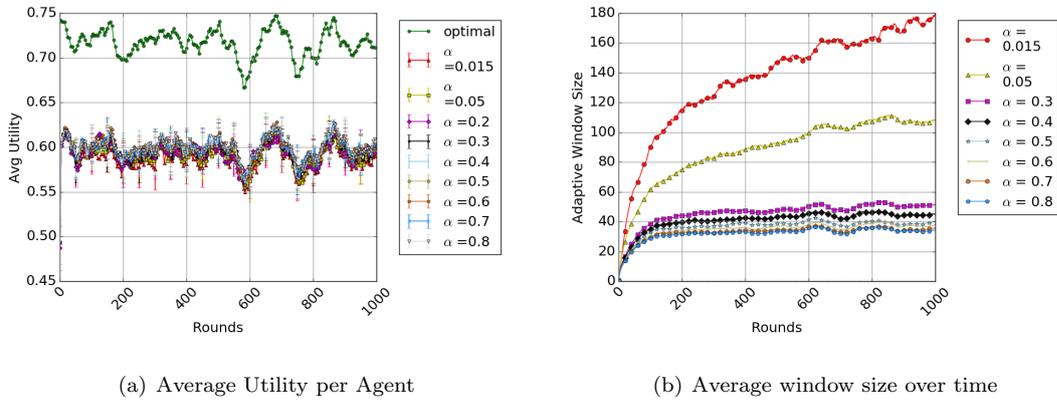


Figure 4 Varying α values in the K-S Test of RaPTaR does not affect its performance.

same experimental parameters as the example. RaPTaR shows a constant average improvement, demonstrating it is robust to dynamic behaviours.

Using different values of forgetting factor or fixed window size does not make these techniques more successful against RaPTaR, as demonstrated in Table 3 where any value of α for RaPTaR performs better. The values of window size and forgetting factor have been chosen based on existing literature, and the intuition that any smaller values are too small to learn accurately from, while larger values would prevent these models from adapting to newer circumstances. One reason these methods do not perform well is because a single forgetting factor or sliding window size will be inappropriate for adapting to every agents' dynamic behaviour.

Varying α has limited effect on RaPTaR's efficacy as seen in both Table 3 and a visualisation of the average utility agents receive over time for different α values in Figure 4(a). Further analysis in Figure 4(b) shows that lower values of α , which require agents to have an extremely high level of confidence to believe there has been a behaviour change, result in more data being retained compared to a higher value of α . Figure 4(b) shows the average window sizes over 100 runs, however in each run the adaptive windows monitoring each group will have varied according to the dynamic situation, which is why RaPTaR outperforms fixed window sizes.

Figure 5 shows that RaPTaR still performs well when behaviours are not repetitive. An example of non-repetitive, random behaviour was depicted in Figure 1(b). However, there is a drop in performance compared to when behaviour is cyclic. This shows that the predictive

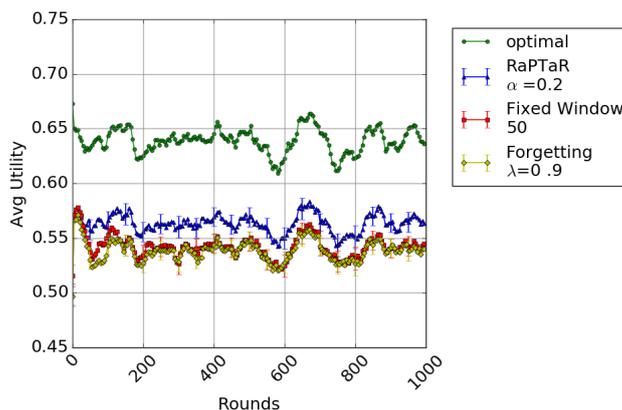


Figure 5 Average utility per agent given random dynamic behaviour in a scale-free network, agents use BRS.

component of RaPTaR is effective, but that improvements could be made by learning when the predictive component is not successful to reduce any inaccurate biased influences it may cause.

Previous work has demonstrated how the accuracy of a trust model to estimate a partner’s exact behaviour is not crucial, but that the trust model should be accurate enough to correctly rank the best agents (Player and Griffiths, 2017). If a group of agents is consistently the best, other techniques which forget at a constant rate will converge on the correct ranking of agents and perform equally compared to RaPTaR. However, with dynamic behaviours, it is unlikely that one group is consistently better than the others over time. Therefore, how consistently accurate the trust assessment is indicates how quickly the trust model adapts to changing situations. The measure of accuracy we use is the root mean squared error (RMSE) at each time point:

$$RMSE = \sqrt{\frac{\sum_{a \in \mathcal{A}} (\tau_{tr,te} - b(t)_{te})^2}{n}} \quad (12)$$

where n is the total number of interactions of all agents in that round, $\tau_{tr,te}$ is the trust estimate between agent tr and trustee te , and $b(t)_{te}$ is the trustee’s true behaviour at time t .

Figure 6 demonstrates that while sliding windows are sufficient in static circumstances, they are not effective with dynamic behaviours. RaPTaR can handle both static and dynamic circumstances by adapting as necessary. Additionally, using an adaptive window means that RaPTaR does not require any prior knowledge of behaviours. Agents in this evaluation and the results from here onwards use DRS instead of BRS, such that interaction outcomes have more granularity. We show that RaPTaR has similar increase in performance, and can handle nominal outcomes as well as binary, which was a motivating factor for using the K-S test for behaviour change detection using interaction outcomes.

The network topology affects agents’ neighbourhood size, limiting the number of available partners. Table 4 shows the mean degree of an agent (i.e. its neighbourhood size) in the respective network topologies¹ and the standard deviation of that mean across all the agents in the network. The standard deviation of the degree in a small-world network is considerably lower than in a scale-free network, so almost all the agents in a small-world network have a selection of 5 to 7 agents to partner with and collect witness reports from. In a scale-free network, most agents have only one or two possible partners, while a small minority of agents will have an extremely high degree. From Table 3, it can be seen that there is little improvement in a scale-free network,

¹Small-world topologies are defined as $n \times n$ lattices in the implementation and are therefore square numbers.

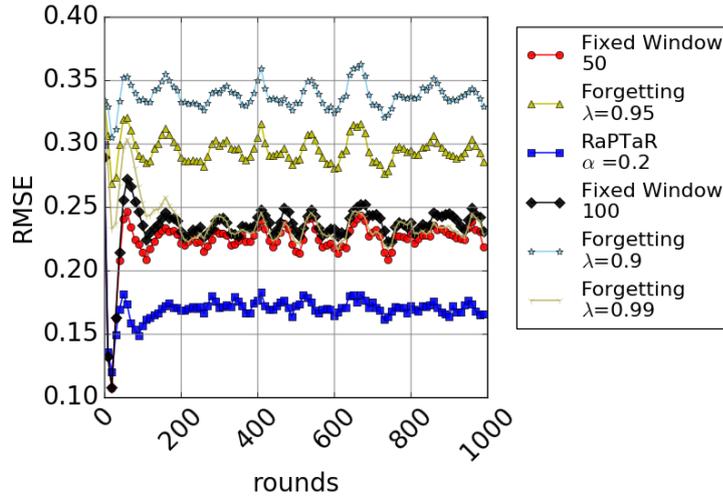


Figure 6 Error in small-world network, where agents use DRS and an exploration rate of 0.2.

because most agents are forced to interact with one of their few neighbours regardless of their behaviour (and therefore also regardless of how accurately they assess trust). A fully connected network best showcases RaPTaR, as agents have a wide variety of choice and it is the assessment technique which most affects an agent’s average outcome. In a fully connected network, we can see that RaPTaR offers the highest improvement, regardless of the trust algorithm it is applied to.

Table 4 Network topology mean degree and standard deviation.

	Total agents in network	Mean Neighbourhood Size (Agent degree)	Standard Deviation
Fully Connected	20	19.0	0.0
	40	39.0	0.0
Small World	100	6.0	1.00995
	196	5.98980	0.92024
	484	6.0	0.99378
Scale Free	100	5.86	4.33594
	200	5.93	5.4877
	500	5.9720	5.86372

In a dynamic environment it is necessary to continually monitor for updates otherwise agents’ views of the world become outdated. However, agents face an exploration-exploitation trade off. Exploration is the probability of interacting with a random partner. The necessary extent of exploration depends on how extreme the changes are. Each run of our experiments generates different dynamic behaviours for each group, and therefore runs can vary from being extremely dynamic to relatively static and there is no knowledge of this in advance. Figure 7 illustrates how different exploration rates affect RaPTaR, sliding windows, and forgetting factors. The results presented are from a fully connected network because agents have a large choice of partners, giving them the opportunity to exploit the knowledge they gain from exploration.

When agents do not explore, as in Figure 7(a), all models are equally blind to changes in the environment and perform poorly. When all models use higher exploration levels, RaPTaR performs the best because it has the capability to learn about the environment and exploit the

knowledge it gains from exploration. Once exploration is as high as 30%, the forced exploration with so many suboptimal partners slightly decreases the average utility of interactions to below 0.6 without having gained any more useful knowledge.

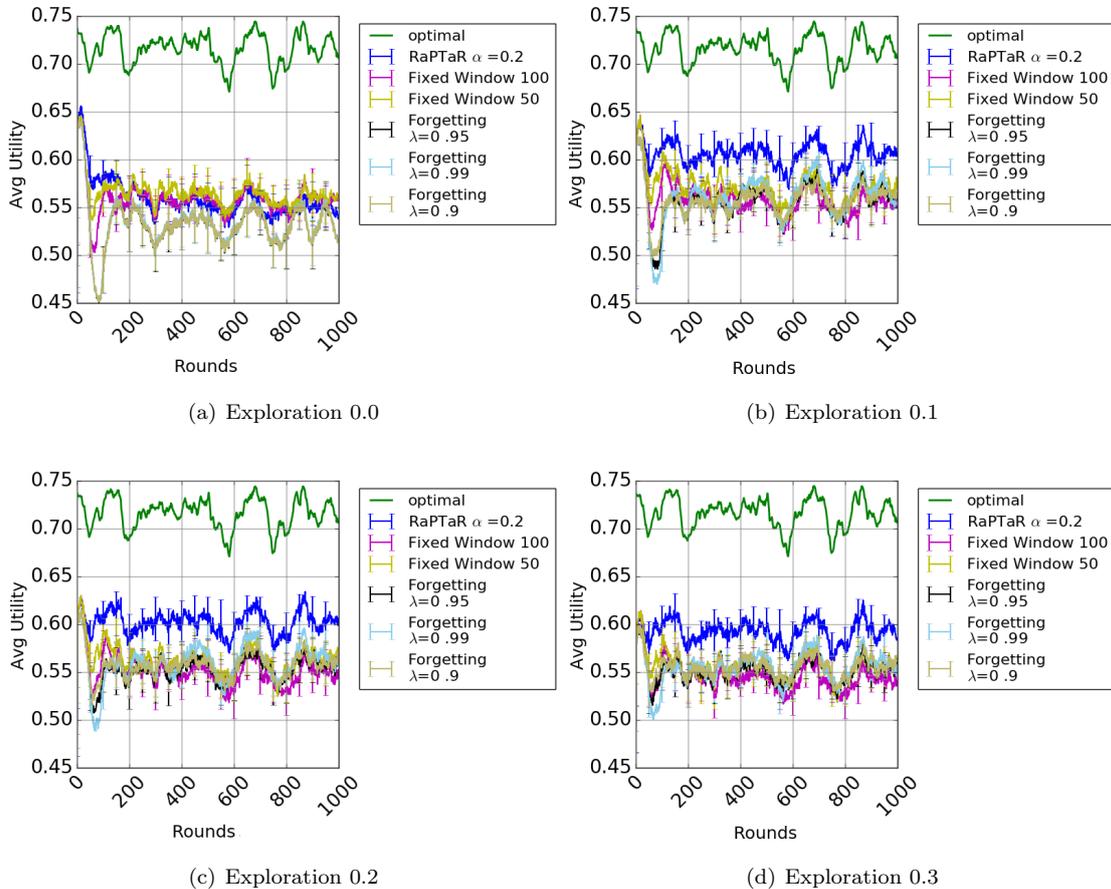


Figure 7 Average utility per agent improves with a small amount of exploration, agents use DRS and are in a fully connected network.

To demonstrate that RaPTaR can be applied to trust models with continuous representations of outcomes, we present results of agents using the TRAVOS trust algorithm in Table 5. One value of forgetting factor, $\lambda = 0.9$, appears to perform equally to RaPTaR however, differences between the averaged results from RaPTaR and a forgetting factor of 0.9 are not statistically significant. Additionally, not all values of λ as a forgetting value are good, and selecting this value would require advance knowledge of the application whereas all values α are comparable. We present fewer results here because TRAVOS is a time consuming trust algorithm due to its reputation collation method. TRAVOS also relies on a large amount of data, and that agents share many interaction partners so that witnesses can provide many reports from which to statistically assess their accuracy as opinion providers. Due to the time constraints from TRAVOS, we present a subset of results in Table 5 using fewer agents and fewer rounds.

6 Discussion and Conclusion

In this paper, we described the necessity for more intelligent data selection techniques in trust and reputation assessment in MAS. We presented RaPTaR, a method which improved trust assessment by reacting to behaviour changes in groups of agents to provide trust and reputation algorithms with past experience data that is statistically assessed to be representative of an

Table 5 Average utility per agents per time step in different network topologies when agents use TRAVOS with either a fixed window size (ws), forgetting factor (λ) or RaPTaR.

Network Type		Fully Connected	Small-World	Scale-Free
$ \mathcal{A} $		20	100	100
ws	50	0.589	0.555	0.543
	100	0.577	0.546	0.536
λ	0.9	0.621	0.586	0.565
	0.95	0.612	0.577	0.558
	0.99	0.584	0.556	0.543
RaPTaR α	0.05	0.613	0.577	0.560
	0.2	0.617	0.579	0.561
	0.3	0.618	0.579	0.562
	0.4	0.619	0.580	0.562
	0.5	0.619	0.581	0.563
	0.6	0.620	0.581	0.563
	0.7	0.621	0.581	0.563
	0.8	0.620	0.582	0.563

agent’s current behaviour. RaPTaR can also learn from the past behaviour of a group to predict an *a priori* estimate of an agent’s behaviour. Our method exploits repetitive behaviour of agents but is also shown to be effective when agent behaviour is dynamic but changing randomly.

Selfish agents cause biased data collection by only interacting with agents they have known to be good in the past. This may mean they only identify and interact with slightly above average partners in a static environment, but when agents have dynamic behaviour this can also prevent trustors from observing improvements in the behaviour of other agents, and ultimately causes suboptimal partner selection. If there is less interaction data collected from a group of agents, it affects RaPTaR’s accuracy to assess the behaviour of that group. Therefore, repeat interactions with partners assessed to be suboptimal must be forced either through the network structure which limits available partners, or through exploration. Exploring to collect data sacrifices a possible higher utility in the short term, however it improves trust assessment in the long term. Exploration is necessary in a dynamic MAS (Carmel and Markovitch, 1999), and this remains true when agents have dynamic behaviour.

The average utility agents receive during a period of static behaviour, or static ranking of the best agents, gives sliding windows and forgetting factors time to identify the best group and then they perform equally as well as RaPTaR. Once dynamic behaviour causes a change in the ranking of agent behaviour, RaPTaR adapts quickly, potentially predicting it, and offers significant improvements for the time period it takes the other models to catch up. This difference can be substantial and is robust, as average results over 100 runs show improvements. The improvements RaPTaR offers on existing methods are statistically significant in a paired test using a p -value less than 0.001, though not statistically significant between itself with different values of the input parameter α , where we have demonstrated throughout the results that there is little dependence on selecting the value for this input.

The computational complexity of RaPTaR scales with the size of an agent’s history of interactions which it must search through after every interaction to statistically detect change using the AdWin algorithm. The K-S test which is undertaken with every split of the window can be done in linear time. Therefore, RaPTaR’s time complexity is $O(W^2)$ where W is the size of the window of instances. The amortized runtime can be reduced using the AdWin 2 algorithm, which uses an exponential histogram data structure to store the window, as opposed to an array as described in this work for illustrative purposes. This algorithm does not require

testing every possible split of W , reducing the number of times the K-S test is performed (Bifet and Gavaldà, 2009). The additional computation RaPTaR performs to learn behaviours when change is detected, is constant. As α decreases, change is detected less frequently, and we saw in Figure 4(b) that this increases the window size. This means that the trust and reputation algorithm the agent is using requires more computation because such algorithms scale with the number of records they have to search through.

6.1 Future Work

RaPTaR's predictive component will not be accurate if behaviour changes are not cyclical. Our results show that RaPTaR's ability to detect and react to random dynamic behaviour is still useful, however, this could be improved upon by integrating error detection and handling, to prevent overfitting to behaviour changes which are not reoccurring. This would make RaPTaR flexible in giving more or less precedence to the predictive component depending on how successful it is.

Exploration can give agents important information, but can also be redundant and costly by interacting with sub-optimal partners. RaPTaR might perform better if agents could adjust the exploration rate in real-time by sensing the level of dynamic change in the environment, similar to WoLF's ability to learn quicker when its performance decreases, and slower while it is performing well (Bowling and Veloso, 2001). Other options include a threshold of minimum trust to veto interactions with agents below the level, or exploring by an algorithm such as simulated annealing.

References

- Gerrit Anders, Hella Seebach, Jan-Philipp Steghöfer, Wolfgang Reif, Elisabeth André, Jörg Hähner, Christian Müller-Schloer, and Theo Ungerer. The social concept of trust as enabler for robustness in open self-organising systems. In *Trustworthy Open Self-Organising Systems*. Springer, 2016.
- Kyle Anderson, SangHyun Lee, and Carol Menassa. Impact of social network type and structure on modeling normative energy use behavior interventions. *Journal of computing in civil engineering*, 28(1), 2013.
- A Bifet and R Gavaldà. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 443–448, 2007.
- Albert Bifet and Ricard Gavaldà. Adaptive learning from evolving data streams. In *International Symposium on Intelligent Data Analysis*. Springer, 2009.
- Michael Bowling and Manuela Veloso. Rational and convergent learning in stochastic games. In *International joint conference on artificial intelligence*, volume 17, pages 1021–1026. Lawrence Erlbaum Associates Ltd, 2001.
- C Burnett, T J Norman, and K Sycara. Bootstrapping trust evaluations through stereotypes. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 241–248, 2010.
- David Carmel and Shaul Markovitch. Exploration strategies for model-based learning in multi-agent systems: Exploration strategies. *Autonomous Agents and Multi-agent systems*, 2(2): 141–172, 1999.
- Cristiano Castelfranchi and Rino Falcone. Principles of trust for mas: Cognitive anatomy, social importance, and quantification. In *Multi Agent Systems, 1998. Proceedings. International conference on*, pages 72–79. IEEE, 1998.

- Ray Chen, Jia Guo, and Fenyue Bao. Trust management for soa-based iot and its applications to service composition. *IEEE Transactions on Services Computing*, 9(3):482–495, 2016.
- Gianni D’Angelo, Salvatore Rampone, and Francesco Palmieri. Developing a trust model for pervasive computing based on apriori association rules learning and bayesian classification. *Soft Computing*, 21(21):6297–6315, 2017.
- Meyerson Debra, Karl E Weick, and Roderick M Kramer. Swift trust and temporary groups. *Trust in organizations: Frontiers of theory and research*, page 166, 1995.
- Karen K Fullam, Tomas B Klos, Guillaume Muller, Jordi Sabater, Andreas Schlosser, Zvi Topol, K Suzanne Barber, Jeffrey S Rosenschein, Laurent Vercoouter, and Marco Voss. A specification of the agent reputation and trust (art) testbed: experimentation and competition for trust in agent societies. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 512–518. ACM, 2005.
- D Gambetta. Can we trust trust? *Trust: Making and breaking cooperative relations*, 13:213–237, 2000.
- Nathan Griffiths. A fuzzy approach to reasoning with trust, distrust and insufficient trust. In *International workshop on cooperative information agents*, pages 360–374. Springer, 2006.
- David Hales and Bruce Edmonds. Evolving social rationality for MAS using tags. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 497–503. ACM, 2003.
- Michael Bonnell Harries, Claude Sammut, and Kim Horn. Extracting hidden context. *Machine Learning*, 32(2):101–126, 1998.
- Pablo Hernandez-Leal, Matthew E Taylor, Benjamin Rosman, L Enrique Sucar, and Enrique Munoz De Cote. Identifying and tracking switching, non-stationary opponents: A bayesian approach. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
- Pablo Hernandez-Leal, Yusen Zhan, Matthew E Taylor, L Enrique Sucar, and Enrique Munoz de Cote. Efficiently detecting switches against non-stationary opponents. *Autonomous Agents and Multi-Agent Systems*, 31(4):767–789, 2017.
- R T Hoens, R Polikar, and N V Chawla. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1(1):89–101, 2012.
- T Dong Huynh and N Jennings. Fire: An integrated trust and reputation model for open multi-agent systems. In *ECAI 2004: 16th European Conference on Artificial Intelligence, August 22-27, 2004, Valencia, Spain: including Prestigious Applicants [sic] of Intelligent Systems (PAIS 2004): proceedings*, volume 110, page 18, 2004.
- A Jøsang and R Ismail. The beta reputation system. *Proceedings of the 15th Bled Electronic Commerce Conference*, 5:2502–2511, 2002.
- Audun Josang and Jochen Haller. Dirichelet reputation systems. In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, pages 112–119. IEEE, 2007.
- Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM, 2003.

- Sarah N Lim Choi Keung and Nathan Griffiths. Trust and reputation. *Agent-based service-oriented computing*, pages 189–224, 2010.
- Matthias Klusch and Andreas Gerber. Dynamic coalition formation among rational agents. *IEEE Intelligent Systems*, (3):42–47, 2002.
- Z Liang and W Shi. Pet: A personalised trust model with reputation and risk evaluation for P2P resource sharing. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, 2005.
- X Liu, A Datta, K Rzdca, and E Lim. Stereotrust: a group based personalized trust model. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 7–16, 2009.
- Xin Liu, Gilles Tredan, and Anwitaman Datta. A generic trust framework for large-scale open systems using machine learning. *Computational Intelligence*, 30(4):700–721, 2014.
- Gehao Lu and Joan Lu. Introduction to the investigating in neural trust and multi agent systems. In *Examining Information Retrieval and Image Processing Paradigms in Multidisciplinary Contexts*, pages 269–273. IGI Global, 2017.
- Doan Tung Nguyen. *Trust Management for Complex Agent Groups*. PhD thesis, Auckland University of Technology, 2017.
- Tung Doan Nguyen and Quan Bai. Accountable individual trust from group reputations in multi-agent systems. In *Pacific Rim International Conference on Artificial Intelligence*, pages 1063–1075, 2014.
- Tung Doan Nguyen and Quan Bai. A dynamic bayesian network approach for agent group trust evaluation. *Computers in Human Behavior*, 2018.
- Caroline Player and Nathan Griffiths. Bootstrapping trust and stereotypes with tags. In *Proceedings of the 19th International workshop on trust in agent societies (Trust at AAMAS)*, 2017.
- Caroline Player and Nathan Griffiths. Addressing concept drift in reputation assessment. In *Proceedings of the 10th International Workshop on Adaptive Learning Agents (ALA @ AAMAS)*, 2018.
- WH Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. Statistical description of data: Are two distributions different. *Numerical Recipes: The Art of Scientific Computing*, pages 730–740, 2007.
- K Regan, P Poupart, and R Cohen. Bayesian reputation modelling in e-marketplaces sensitive to subjectivity, deception and change. In *Proceedings of the National Conference on Artificial Intelligence*, 2006.
- P Resnick, K Kuwabara, R Zeckhauser, and E Friedman. Reputation systems. *Communications of the ACM*, 43(12), 2000.
- Amirali Salehi-Abari and Tony White. Dart: a distributed analysis of reputation and trust framework. *Computational Intelligence*, 28(4):642–682, 2012.
- M Sensoy, B Yilmaz, and T J Norman. Stage: Stereotypical trust assessment through graph extraction. *Computational Intelligence*, 32(1):72–101, 2016.
- Mudhakar Srivasta, Li Xiong, and Ling Liu. Trustguard: countering vulnerabilities in reputation management for decentralised overlay networks. In *Proceedings of the 14th international conference on world wide web*, pages 422–431, 2005.

- Ugur Eray Tahta, Sevil Sen, and Ahmet Burak Can. Gentrust: A genetic trust management model for peer to peer systems. *Applied Soft Computing*, (34):693–704, 2015.
- Phillip Taylor, Lina Barakat, Simon Miles, and Nathan Griffiths. Reputation assessment: a review and unifying abstraction. *The Knowledge Engineering Review*, 33, 2018.
- L Teacy, J Patel, N Jennings, and M Luck. Travos: Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems*, 12(2):183–198, 2006.
- L Teacy, M Luck, A Rogers, and N Jennings. An efficient and versatile approach to trust and reputation using hierarchical bayesian modelling. *Artificial Intelligence*, 193:149–185, 2012.
- Giulia Traverso, Carlos Garcia Cordero, Mehrdad Nojournian, Reza Azarderakhsh, Denise Demirel, Sheikh Mahbub Habib, and Johannes Buchmann. Evidence-based trust mechanism using clustering algorithms for distributed storage systems. *IACR Cryptology ePrint Archive*, 2017.
- Alexey Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106(2), 2004.
- Geoffrey I Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4):964–994, 2016.
- Ying Yang Xindong Wu and Xingquan Zhu. Combining proactive and reactive predictions for data streams. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge Discovery in Data Mining*, pages 710–715. ACM, 2005.
- L Xiong and L Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE transactions on knowledge and data engineering*, 16(7):843–857, 2004.