AFRL-AFOSR-UK-TR-2018-0042

Engineering Emergence in Large-Scale Simulations

Simon Miles
KING'S COLLEGE LONDON
THE STRAND
LONDON, WC2R 2LS
GB

11/02/2018
Final Report

Air Force Research Laboratory
Air Force Office of Scientific Research
European Office of Aerospace Research and Development
Unit 4515 Box 14, APO AE 09421

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
02-11-2018	Final	01 May 2015 to 30 Apr 2018

4. TITLE AND SUBTITLE
Engineering Emergence in Large-Scale Simulations

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA9550-15-1-0092

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)
Simon Miles

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
KING'S COLLEGE LONDON
THE STRAND
LONDON, WC2R 2LS GB

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
EOARD
Unit 4515
APO AE 09421-4515

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/AFOSR IOE

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-AFOSR-UK-TR-2018-0042

12. DISTRIBUTION/AVAILABILITY STATEMENT
A DISTRIBUTION UNLIMITED: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The PI examined emergent behaviors in large multi-agent systems. Specifically, the PI worked to understand, identify, and experiment with emergent system behavioral patterns. This work is important for understanding modern complex systems, and necessary for systems to positively responding to these behaviors whether planned or unplanned action or reaction. The work produced one journal paper and two conference papers, and was useful for advancing the state of the art in the area of complex systems/networks.

15. SUBJECT TERMS
Emergent behavior, Normative reasoning, Multi-agent systems, Machine intelligence, EOARD

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			MAILLOUX, LOGAN
Unclassified	Unclassified	Unclassified	SAR		19b. TELEPHONE NUMBER *(Include area code)* 314 235 6163

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Engineering Emergence in Large Scale Simulations: Final report (May 2015 - April 2018)

PI: Dr Simon Miles

Grant Number: FA9550-15-1-0092

Contents

List of Figures

1 Summary

The Engineering Emergence in Large-scale Simulations (EELS) project investigated the phenomenon of emergent patterns of behaviour (norms) in complex multi-agent systems, and develops mechanisms to engineer such systems to recognise norm emergence, and encourage the adoption of norms beneficial to the system. We undertook and published a thorough review of the state of the art pertaining to engineering the emergence of norms, which highlighted gaps to be addressed as part of our project. On this basis, we developed a formal model of emergent norms, allowing us to then specify mechanisms to,

first, detect the emergence of norms within a large-scale system in which many events are occurring and, second, reason about the expectations created by the emergence of norms. We then moved on to engineering beneficial norm emergence in resource-constrained systems through multiple mechanisms applicable to different systems: a protocol for mutual commitment of agents to beneficial emergent norms, a probationary mechanism to limit agents until it is known whether they fulfill these commitments to agreed norms, and resource-bounded explicit punishment to ensure compliance.

2 Introduction

A multi-agent system consists of multiple autonomous software components (agents) interacting to perform tasks, either cooperatively as a group, or competitively as individuals. As agents within a complex multi-agent system interact with each other and their environment, stable patterns of macro-level behaviour can arise. This *emergent* behaviour may not be predictable by the designers, or the agents themselves, but can have either beneficial or detrimental effects upon the system utility.

Emergent behaviour exhibits the following characteristics. It is behaviour that becomes recurrent across agents and time during the lifetime of the system. It is not expected from the micro-level design or prior knowledge of the agents or their environment but instead arises from the interactions of those agents and the environment. And it is behaviour that has some stability due to influences within the system rather than just being observable at some time by coincidence.

One category of emergent behaviour is *social norms*. Social norms are customary rules that govern behaviour in groups. They can aid cooperation and coordination by describing expected behaviour, and so reduce reasoning in common situations. Violation of norms by individual agents is possible, and sometimes desirable, but it risks punishment, either through explicit action by other agents or implicitly through loss of utility. As such, we consider norms to be emergent behaviour that *matters* for the system's function. That is, because violation of norms has an effect on the violating agent and on the system as a whole, it can be evaluated in terms of its benefit and also enforced. Given the focus of the project on engineering emergence, we have therefore focused on social norms as emergent behaviour.

The goal of the EELS project was to explore the engineering of emergent norms arising in complex multi-agent systems in order to provide a benefit to the system. By *engineering* we mean the harnessing or control of emergent behaviour so that it can be directed in a way that benefits the system. Engineering norms means propagating or collapsing norms and can be achieved by multiple means. If a norm is harmful, more compliance means more harm, while if a norm is beneficial, more compliance means more benefit. More compliance to a norm also means that behaviour can be expected, and so relied on, which allow for further benefits in itself. We were also concerned with the practical limitations placed on mechanisms for engineering emergence within multi-agent systems of *large scale*. We assume that resources are naturally bounded, and that agents must be trusted to comply with beneficial norms at some points.

As detailed in the sections below, we achieved the following in this project. First, we undertook and published a thorough review of the state of the art

pertaining to engineering the emergence of norms. On this basis, we developed a formal model of emergent norms, allowing us to then specify mechanisms to, first, detect the emergence of norms within a large-scale system in which many events are occurring and, second, reason about the expectations created by the emergence of norms. We then moved on to engineering beneficial norm emergence in resource-constrained systems through multiple mechanisms applicable to different forms of system: a protocol for mutual commitment of agents to beneficial emergent norms, a probationary mechanism to limit agents until it is known whether they fulfill these commitments to agreed norms, and resource-bounded explicit punishment to ensure compliance.

3 Methods, Assumptions and Procedures

Our methods involved review of the state of the art, development of models, mechanisms and protocols, and testing of the latter through simulation. Each step tackled a part of the problem of how agents within a system can detect the emergence of norms, evaluate them, use them to their benefit, and ensure the persistence of those norms evaluated to be of most benefit.

A key practical assumption of the work is that systems are distributed and bounded. This means that, first, there is no external observer that can view the whole system, detect emergence, evaluate behaviour, and enforce. These operations will have to be performed by the distributed agents, requiring communication and limited viewpoints to be accounted for. It also means that the resources brought to reasoning and enforcement are not unlimited, meaning that agents may sometimes need to trust each other to comply without monitoring or limit the amount of enforcement actions taken.

4 Results and Discussion

The project results can be categorised into understanding the state of the art (Section 4.1), methods for detecting and reasoning about emergence of norms (Sections 4.2, and 4.3), and methods for engineering the system to adopt and comply with norms considered beneficial while accounting for constrained resources (Sections 4.4, 4.5 and 4.6).

4.1 Survey of the field

A survey paper has been published in the Knowledge Engineering Review [11]. The survey identified three distinct stages in the engineering of emergent norms: detection of emergent behaviour, evaluation of that behaviour, and its encouragement or discouragement in the agent population. Existing work has focused on detection and evaluation of emergent norms from either an individual agent perspective, or via a system wide approach. Likewise, work on the commitment to norms (as part of encouraging the spread of behaviour through a population) has focused upon individual commitment by agents based upon their own evaluation of the norm's value.

The literature review identified several gaps in the state of the art that were, in part, addressed by this project, specifically: the lack of a distributed emergent norm identification mechanism, the lack of a distributed norm evaluation

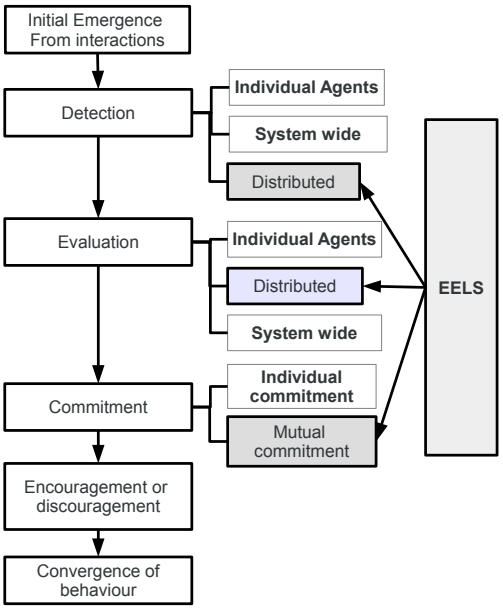

Figure 1: EELS in the research landscape

mechanism, and the lack of a mutual norm commitment protocol. Figure 1 shows a conceptual view of how EELS fits into the existing research landscape.

4.2 Detection of norm emergence

Emergent behaviour arising in a system from autonomous agents interacting with each other, and the environment, will manifest as observable patterns of related *events* repeated over space and time. These events are state changes either caused by agent actions or the environment, and the relations between them can be characterised as constraints. The constraints could be spatial, temporal, or expressed in terms of other properties of the events (such as the agents performing the actions). These repeating patterns of events can be represented as *complex events*, and, using our event-based formalism, we can specify such a complex event in terms of two or more simple events and the relationship between them.

Since we are looking for emergent social behaviour (norms) we are interested in complex events where later events (e.g. those deriving from agent actions) are in some way *caused* by previous events. If, for example, the complex event, $ce \equiv e_1 \prec e_2$ is the manifestation of normative behaviour then we would expect e_2 to occur due to an agent following a norm that obliges it to bring about e_2 in response to e_1, or, in other words, there is some kind of *causal relationship* between e_1 and e_2.

Figure 2 shows a conceptual view of our norm detection process. The complex event specification, or *complex event signature* (CES), can be derived in two ways: first, from domain knowledge, and second, from data mining the logs of system events. In the first approach, domain knowledge is used to create

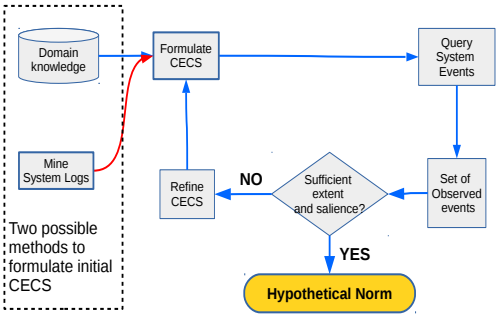

Figure 2: Conceptual view of our norm detection approach

a hypothetical causal pattern (or causality graph) concerning an intervention (some change to the system) and a possible result of that intervention. In other words we make a guess at what will happen. This causality graph will be made up of observable events linked by relationship constraints, and so will make up a CES. The CES is used as a query across the system events. These system events are either from the real system, or from simulation traces, and may either be examined as a set of logs held in a database, or as an incoming stream of events. The result of this query will be a set of observed complex events. The properties of the complex events within this set will allow us to refine the CES and re-query if necessary. Once sufficiently refined, the CES can be translated to a formal representation of a norm.

The second approach begins with a record of the system events themselves. This is more exploratory than the first approach, since no *a priori* hypothesis is required. Data mining techniques are used to extract complex events (sequences of events linked by temporal and other constraints) related to the area of interest, e.g. a specific process or metric. These extracted observed complex events can be refined. The observed complex events, along with domain knowledge, are then used to formulate a CES. This can be subsequently used as a query and further refined, as in the first approach, and then translated to a formal representation of a norm.

In both approaches, the refinement process is similar. If no complex events matching the specification are found then we refine the specification: either changing the events, or loosening the constraints. If complex events matching the specification are detected in the system, we can then examine the occurrences to determine the degree to which the simple events that make up the complex event are causally related, i.e. that an event that forms part of a complex event is caused (in some way) by an earlier event, and itself causes a subsequent event. We therefore seek to distinguish these causal relationships from mere correlation, such as two events happening in a temporal sequence because of agents following independent plans and timetables. We use a probabilistic determination of causality, so we use a threshold approach — if the probability is below a certain threshold then we again refine the complex event specification; if it is above the threshold then we make the assumption that the relationship is causal.

We can also refine the specification by examining the *extent* and the *salience*

of the hypothetical norm in the population at each given time. The extent is the apparent prevalence of the norm: how many agents in a given population are following the norm? The salience is the apparent strength of the norm among individuals who ever perform the normative behaviour: how often do agents following the norm comply with it?

If a complex event is above the threshold, we then perform an inference step where we seek to infer a norm (or set of norms) from the events that make up the complex event. This inference is guided by the complex event specification, including the relationship constraints, and knowledge, if any, about the goals and beliefs of the agents performing the actions involved in the complex event.

Algorithm 1 gives our overall approach. We generate a possible CES from domain knowledge, recorded events (via data mining) or both (line 1). The loop (lines 3 — 27) is repeated until the CES is refined and a norm found. Matching events in the event store are found in line 4, using the CES. Causality is checked in line 5, and the CES refined and the loop restarted if the events are deemed not to be causally related (lines 6-9). Similarly, significance is determined in line 10, and inevitability in line 15, with the CES refined and the loop restarted if these tests are failed. In line 20, the extent and salience are calculated. If they exceed the thresholds a hypothetical norm is generated (line 22) and the refine flag is set to false, so the looping ends, otherwise the CES is refined (line 25).

The change of extent and salience over time can give an insight into the degree to which the norm has emerged in the population (or sub-population) of agents, and the manner in which it is spreading. An emerging norm that originates in a small group or from a norm 'entrepreneur' [8], will start with a very low extent but a high salience. Over time, if the norm spreads, the extent will increase while the salience may go down as non-committed agents try out the norm. If the norm is adopted by these agents the salience will then increase again. An emerging norm that arises from a population of agents learning the best way to perform some social task from a number of possibilities may start with a relatively high extent, but a low salience, as each agent tries out different behaviour (including that complying with the hypothetical norm). Over time, if agents converge on one type of behaviour, both extent and salience will rise. Given these different characteristics of emerging (as opposed to established) norms, we can set thresholds for extent and salience appropriately in order to detect norms before they are fully established.

We evaluated the detection mechanism using our multi-agent testbed. We set up a scenario where sub-populations of agents would develop different norms. As an example of our results, Figure 3 shows the extent of one norm in three populations: all agents, Group 1 (where the norm emerged), and Group 2 (a neighbouring group). Figure 4 shows the average salience in the same groups. With an appropriate threshold for extent and salience, we can refine the target of the norm to Group 1, since only they have both high values for both. We can also see that the average salience rises consistently and quickly in Group 1 — this indicates that the agents in that group who first start performing the normative behaviour tend to perform it consistently (i.e. they continue to perform it) even when many of their group are not. This is indicative of an emerging norm which starts with a core group of adopters and then spreads throughout the wider population.

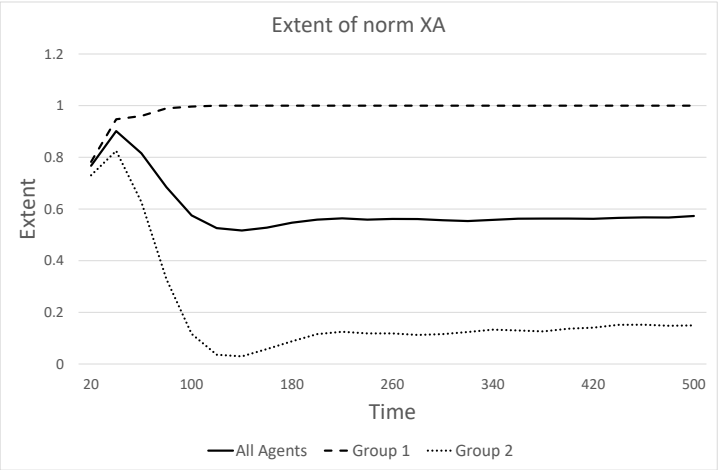

Figure 3: Extent of emerging norm over time in different agent groups

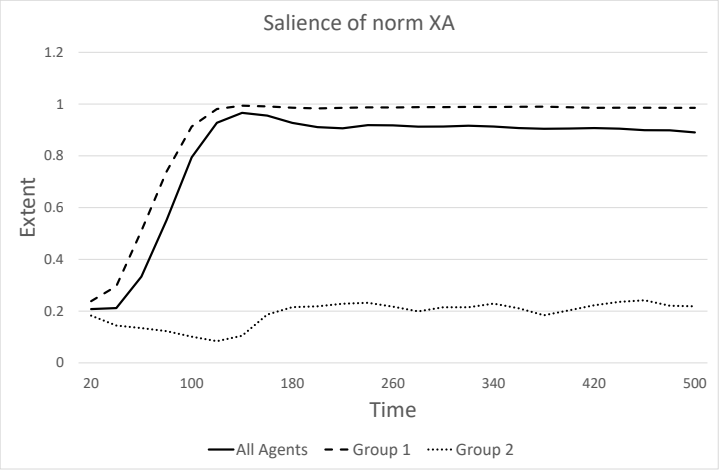

Figure 4: Salience of emerging norm over time in different agent groups

4.3 Expectation-based decision making

Norms are expressions of *expected* social behaviour [2], so they will create expectations. If we are aware of a formal norm obliging a driver to stop at a red light, then we will expect a vehicle to stop even if we have not previously observed nine other vehicles doing so. As our knowledge of social norms changes (either due to experience or education), our expectations will also change. For example, if we move to a country with different traffic laws our expectations will change.

In exploring expectation-based reasoning, we followed Castelfranchi et al. [4] in considering expectations to be more than simple predictions or beliefs about the future, but instead to be beliefs about the future that one is subjectively interested in either becoming true or false. This view of expectations follows both the natural sense of an expectation being something that an agent is "waiting for", and also has the practical benefit of limiting the number of expectations that a software agent has to reason about.

An *abstract expectation* is one not grounded by a specific situation. For example, "cars will stop at red traffic lights", "my colleague will not be late for meetings", "heavy grey clouds mean it will rain soon" are all expressions of abstract expectations. Abstract expectations are added or removed due to three factors: experience, norms, or communication with others. As we observe the behaviour of others (either individuals or groups) it is natural to assume that past behaviour will give an insight into future behaviour. This is the essence of creating expectations via experience. If we observe nine vehicles stopping at a red light, we may reasonably expect the tenth to also stop. Likewise, if we see consistent examples of behaviour that contradict our expectations we may modify those expectations.

Communication can also create abstract expectations. If someone we trust tells us that they always stop at red lights, then this will create an expectation that they will do so, regardless of our knowledge of norms or previous experiences. As before, communications that contradict previous expectations will modify them, as will changes in our level of trust in the communicator. If we find out that the driver has frequently ignored red lights, we may change our expectations of their future behaviour.

As well as abstract expectations, there are more specific expectations that are grounded in a actual situations that have occurred. We call these *active expectations*. They can be seen as instantiations of abstract expectations. For example, "that car will stop at that red traffic light", "my colleague John will come to my office on time for our 3 o'clock meeting", "there are big dark clouds in the sky, so it will rain soon", are all active expectations. Active expectations are triggered from abstract expectations by specific events, such as a traffic light turning red, inviting John to a meeting, or seeing storm clouds. If an agent does not perceive the triggering event, then no active expectation is instantiated — if it does not know that a light has turned red then it will not expect the approaching car to stop. Active expectations may be discarded if it becomes clear that they can never be satisfied. If I see a car drive through a red light, I will no longer expect it to stop. However, other cases may be more vague, with a broader range of possible fulfilment conditions. If I am under heavy storm clouds, I may well expect it to rain until the clouds move away, even if it does not actually rain.

Although agents will only hold active expectations where they have observed

a triggering event and they have an interest in the fulfilment, or non-fulfilment of the expectation, they may hold abstract expectations about events that they are not currently interested in. For example, while sitting in my office, I have no interest in whether cars stop at red lights or not, but I know that they generally do so and it is useful to retain this abstract expectation for the next time I wish to cross a road.

Expectations are not all equally strong. Instead, each expectation is held with a degree of certainty [9]. Some expectations may be very strong, to the extent that it would be a surprise if they are not fulfilled, but others may be more akin to "rules of thumb" or stereotypes where exceptions to the rule (where expectations are not fulfilled) are unusual but not unexpected to the same degree. Therefore, our conception of expectations includes the notion of a *weight* to represent this degree of certainty. This weight may be derived from experience, or the degree of trust in the source of the expectation. It is rarely a fixed quantity, unless the expectation is derived from axiomatic articles of faith. The degree of certainty of an expectation may be eroded if it is not fulfilled.

We evaluate a model of expectation-based reasoning using a simple multi-agent system consisting of agents performing abstract tasks and helping neighbours in their tasks. Each agent has a finite number of possible actions and each task consists of performing a series of actions in a specific order. Once a task is completed, the agent chooses a new task. There are a finite number of different tasks and each agent has a probability function that determines which one it will select. Each agent has a different probability function. As well as performing an action on its own task, each agent also performs one help action for a neighbour every time step. To successfully help a neighbour it must correctly determine which action that neighbour is going to perform next. Agents can see the previous actions performed on a task and their expectations of the neighbour's behaviour guide their choice.

For example, the set of possible actions are A, B, C and agent, a_1 is working on a task sequence (A, B, B, C). It has performed two actions, (A, B). Its neighbours, a_2 and a_3, try to support it — a_2 predicts action C and a_3 predicts action B. Since a_3 is correct it receives a reward for helping, a_2 receives nothing.

In our simulation, agents generated abstract expectations by observing when a neighbour completes a task. The abstract expectations are generated for individuals, so observing one neighbour complete a task will not lead an agent to have any expectations about a different agent. The initial weight is set to 0.5. Fulfilled expectations increase the weight by 0.05, expired expectations decrease the weight by 0.05. Abstract expectations are discarded if the weight falls to zero. Over time, if an agent prefers some tasks to others, its neighbours will build up strong expectations about its behaviour.

We used three scenarios that represent agents with a strong, medium or weak preference for one task. In each scenario each agent chooses from an individual, random, subset of eight tasks (out of a possible 20), but has a higher probability to perform some than others. Table 1 shows the probability functions for each scenario. So, for example, in a strong preference scenario each agent will choose one task with an 0.8 probability, a further two tasks each with a 0.05 probability, another five with a 0.02 probability and it will never select the other 12 possible tasks. This probability function is fixed at the start of each run.

We compared our expectations module with two other methods: a probabilistic selector and a simple learning algorithm. The probabilistic selector com-

Scenario	Probabilities for task selection
strong	0.80, 0.05, 0.05, 0.02, 0.02, 0.02, 0.02, 0.02
medium	0.51, 0.07, 0.07, 0.07, 0.07, 0.07, 0.07, 0.07
weak	0.40, 0.10, 0.10, 0.10, 0.10, 0.07, 0.07, 0.06

Table 1: Scenario probability distribution for task selection

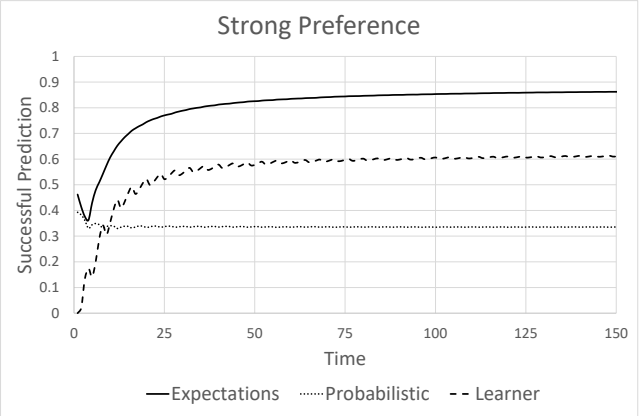

Figure 5: Proportion of successful predictions in strong preference scenario

pares the current sequence of actions to the full set of known task sequences. It creates a set of possible tasks that the agents is working on, selects one at random and uses that to predict the next action. For example, if the set of known tasks is $\{(A, B, C), (A, B, B), (A, C, B), (A, C, C)\}$ and the current sequence is (A, B), then the possible tasks are (A, B, C) and (A, B, B) and the agent will select one at random, leading them to predict the next action as either C or B. In the simple learning algorithm approach, each agent builds up a set of observed sequences of events (for example, A followed by B) performed by each of its neighbours along with a count of the number of times the neighbour has performed that sequence. When it sees a neighbour's action it then predicts the next based upon the sequence with the highest count.

Figures 5, 6, and 7 show the proportion of support actions that were successful (i.e. where the agent successfully predicted which action its neighbour would take) over time for each approach. For agents using the probabilistic selector, the scenario does not matter, so only a single line is shown. It can be seen that the agents using expectations are rapidly able to predict with some accuracy the behaviour of their neighbours, even from a starting point with no expectations. The simple learning approach also becomes more successful over time, but is never as successful as the expectations method.

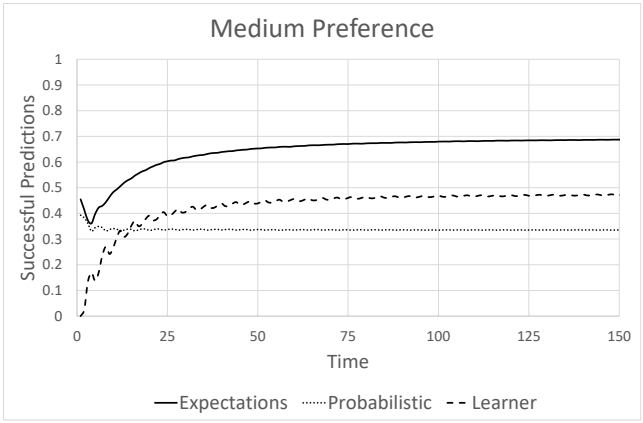

Figure 6: Proportion of successful predictions in medium preference scenario

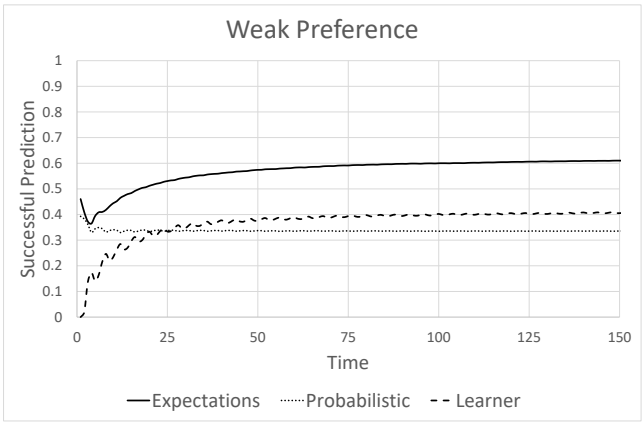

Figure 7: Proportion of successful predictions in weak preference scenario

4.4 Mutual commitment to emergent norms

As the last section showed, there is a value to certainty in your expectations of others' behaviour because you do not have to pay costs for preparing for the *contingency* of the others behaving in any of multiple ways. If a de-facto norm becomes widespread enough, you can rely on it, but this takes time during which contingency costs apply. If everyone commits to the behaviour, this shortcuts the process.

Emergent behaviour is not designed into the system from the start, but in a complex system, new behaviour can emerge and it can be beneficial. If a behaviour emerges, then the system needs to spot it and evaluate it before commitment can take place. Practical distributed systems are decentralised, so we need detection, evaluation and commitment to themselves be decentralised.

To achieve mutual commitment to emergent behaviour, two roles are required: *trend-spotters* observe emergent behaviour of potential interest using an approach such as that described in Section 4.2, and *coordinators* enact a mutual evaluation and commitment protocol. Trend-spotter agents monitor the behaviour of their neighbours and maintain a set of expectations about that behaviour in order to detect behaviour that is increasing in prevalence, *up-trends*. An *up-trend* occurs when the expectation weight of a behaviour increases by more than the threshold over a specified time period that exceeds a threshold. The time period and threshold should be chosen to reduce the risks of temporary fluctuations in expectation weight being counted as up-trends. Coordinators aggregate information about these up-trends and are responsible for eliciting evaluations from all members in the group, assessing whether to commit to new behaviour, and informing the group of the commitment decision.

Figure 8 shows a conceptual view of the protocol for mutual commitment. Table 2 gives a summary of the message types used in the protocol.

Table 2: Summary of mutual commit protocol messages

Message	Sender	Rec'd	Purpose
INFORM	Spotters	Coord	Inform about behavioural up-trends
REQEVAL	Coord	All	Request evaluation of behaviour
EVAL	All	Coord	Inform about value of behaviour
COMMIT	Coord	All	Inform of decision to commit to behaviour

Upon detecting one or more up-trends, each trend-spotter reports to the coordinator agent by sending an *information* message:

$$INFORM(\{exp_1, ..., exp_n\})$$

where exp_i is an expectations that is up-trending. If there is more than one trend-spotter, the up-trends are aggregated and ordered by the number of trend-spotters seeing the specific up-trend.

The coordinator sends a *request for evaluation* message to each agent in the group:

$$REQEVAL(\{exp_1, ..., exp_n\})$$

where exp_i is an expectation to be evaluated.

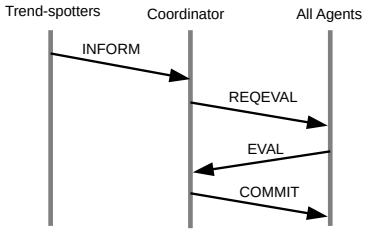

Figure 8: A conceptual view of the mutual commitment protocol

Upon receiving a REQEVAL message, each agent evaluates the behaviour represented by each expectation. We assume here that agents are able to evaluate behaviour and proposed behaviour using a utilitarian approach with some agreed objective metric amongst the group. If this is not possible then the approach can be modified to allow agents a simple vote whether to commit to the behaviour or not. For example, this allows the approach to be used where behaviour is evaluated via ordered preference rather than measurable utility values. The evaluation is not done on the basis of the *current* normative environment, but on a hypothetical one where *every* agent performs the evaluated behaviour. In this way, the evaluation can account for any benefits of coordination and cooperation. For example, when evaluating the benefit of driving on the right side of the road, an agent will assume that every vehicle will drive on the right side of the road, even if everyone currently drives on the left.

Each agent sends an *evaluation* message back to the coordinator:

$$EVAL(\{ev_1, ..., ev_n\}, val_{current})$$

where $ev_i = \langle exp_i, val_i \rangle$ is a tuple containing an expectation, exp_i and its evaluated benefit, val_i, and $val_{current}$ is the evaluated benefit of the agent's current behaviour.

The coordinator makes its decision based on the aggregate evaluations and a comparison with the evaluation of the agents' current behaviour. It comes to one of two conclusions: commit to one behaviour, or make no commitment. If no option has an aggregate value above that over the current behaviour, then no commitment will be made, otherwise it will choose to commit to the option with the highest aggregate value. The coordinator will send a *decision to commit* message to each agent:

$$COMMIT(ev_{ins}, ev_{norm})$$

where ev_{ins} i the instigating event and a ev_{norm} is the normative event ev_{norm}. The message can also be empty to indicate no commitment.

On receipt of a non-empty *COMMIT* message, each agent makes the specified commitment. This means that when acting in response to an instigating event, the agent will bring about the normative event. What this means in practice depends upon the agent architecture and reasoning process. The agent may adopt a norm that obligates the behaviour, or it may add a goal to bring about the normative event when it observes an instigating event.

Once a group commits to a certain behaviour, does this bind the group to that behaviour forever? Clearly, such inflexibility could be detrimental to the group. Circumstances may change that reduce the value of the behaviour, such as new agent abilities or goals, or environmental changes that make the commitment harmful. Also, the initial evaluation of the behaviour may prove to be incorrect. It may be hard for agents to correctly evaluate the behaviour — they may be working with incomplete knowledge, or there may be detrimental effects that only manifest after some time. Therefore, a group requires some way to question whether existing commitments should be revoked. One possibility is a simple regular re-evaluation of each commitment after a set time period. If the value of the behaviour is found to have dropped by more than a specified threshold, either due to changes in the system or new information, then the commitment is dropped by the entire group.

Commitment has a cost because it limits *exploration* to find better untried alternative behaviours in future. Its value depends on the size of contingency costs. Our experiments showed that, as expected, with low contingency costs, commitment is detrimental, but becomes of increasing value as contingency costs increase.

The results show that, as one would expect, the trend-spotting approach does lead to the earlier adoption of a single norm compared to allowing the organic spread of the norm. However, when there is no contingency cost and therefore no benefit to certain coordination, the approach leads to a worse eventual utility. This is because where there is little or no benefit, new norms that are only marginally better than the initial strategy may spread through the population and be detected by the trend-spotter, and then committed to by the population. In contrast, where the contingency cost is high, there is a large benefit to certain coordination and marginal improvements will not spread. In this case the trend-spotters are unlikely to detect these marginal strategies, while strategies which are greatly superior will still spread. Figure 9 shows the benefit of the trend-spotting approach using a single trend-spotter with different contingency costs on the summed total utility over 250 time-steps. The drop-off at 0.5 contingency cost is due to the nature of the scenario, where the strategy distance from the ideal strategy is constrained

With respect to the type of spotter, the results shown in Figure 10 that selecting on the basis of degree centrality in the agents' social network is superior to random spotters, although this effect is likely accentuated by the scale-free nature of the network and hence the existence of hub nodes with many neighbours. Increasing the number of spotters was shown to increase the average utility of the adopted norm when using randomly selected spotters (Figure 11).

4.5 Probationary restrictions to ensure norm compliance

Mutual commitment works by assuming that agents can be trusted to obey their commitments, at least in sufficient numbers and for sufficient time for reduced contingency costs to exceed any lack of benefit from exploration. Where resources are constrained, such trust may be more feasible than monitoring of the whole system. In a system with structured authority, we can go one step further and try to restrict agents rights to act until they can be trusted without restriction.

There are two norm-based approaches to controlling the behaviour of agents:

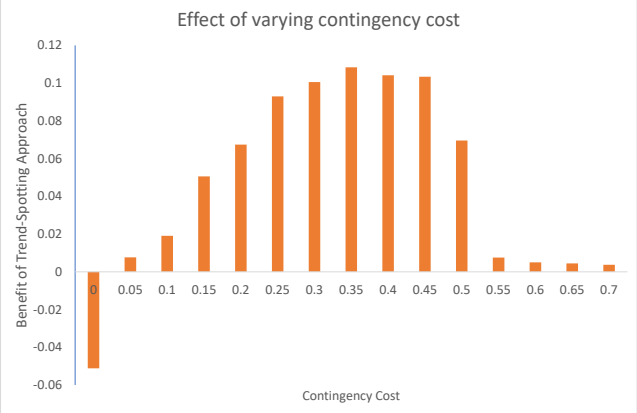

Figure 9: The effect of varying contingency cost on commit benefit

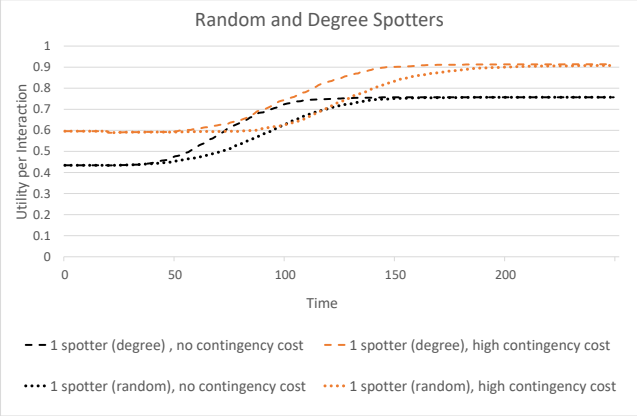

Figure 10: Comparing degree centrality spotters with random spotters

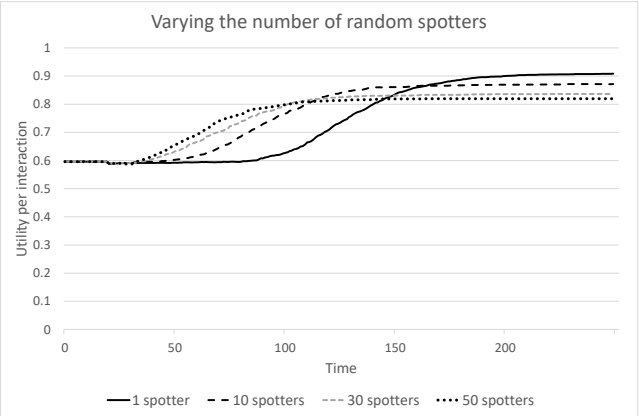

Figure 11: The effect of varying number of random spotters (high contingency cost)

regulation and *enforcement*. In a regulation approach, such as that embodied by *electronic institutions* [6], the system itself imposes constraints on what an agent can do, but this severely restricts autonomy, and thus reduces system flexibility. In contrast, an enforcement approach specifies what an agent ought to do (or not do) and relies on *enforcement* of the rules so that violators are punished. This allows agents to retain their autonomy, but it also requires an enforcement mechanism to monitor agent behaviour for norm violations in order to motivate agent compliance.

Of course, in this latter enforcement approach, agents must be informed of the applicable norms, so that they can comply with them, and one way of doing this is through formalising norms as *contracts* and ensuring that agents are aware of them. Then, activity within a system can be made contingent on agreeing to a contract, in the same way that using a service can require agreeing to a set of terms and conditions. As suggested by Modgil et al. [17], a *contract* is a set of *clauses* describing the normative behaviour expected of given agents, the *contract parties*, where a contract clause specifies an obligation, prohibition or permission. Contracts are formal expressions of expectations that explicitly detail what behaviour is expected, by whom, and under what circumstances. The penalties for breaking a contract are also usually explicit, either within the contract itself, or in the body of regulations that govern contracts in general. Contract parties can thus determine the actions they must perform, and others can clarify what constitutes a violation.

Yet, just as in human societies, in this approach computational autonomous agents may choose to violate norms even after agreeing to comply with a contract specifying them. Such non-compliance can be a risk to the system or organisation in which these agents operate, the extent of the risk depending on the likelihood of an agent failing to perform its assigned task and the consequence of such a failure. While agents can be monitored and norm violations may be detected, this detection may not prevent the consequences of the norm

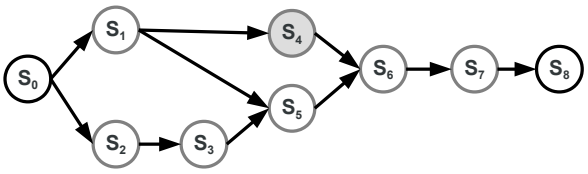

Figure 12: A state graph for a task with an undesirable state, S_4

violation; for example, if the violation is due to an atomic action that can only be detected once it has been performed then monitoring cannot prevent the consequences of the violation.

In response, we addressed the risk involved in non-compliance with norms and contracts by proposing the use of *probationary contracts* to reduce the consequences of uncertain trust in normative multi-agent systems [12]. Probationary contracts are special contracts that include extra conditions or norms to restrict the behaviour of an agent performing the probationary role that commits to the contract. To do this, we simply insert an additional sub-goal (or intermediate goal state) that must be achieved during a task in order to restrict a choice between paths.

For example, Figure 12 shows a task expressed in a directed acyclic graph (DAG), with the nodes representing environment states and the edges representing transitions caused by agent actions. An agent begins in state S_0, and the task contract requires achieving the goal of reaching S_8 (which we could also consider as a norm), while complying with the prohibition against bringing about state S_4. These should ensure that an ideal agent will choose a path to S_8 that does not pass through S_4. However, for selfish reasons a non-ideal agent may choose to violate the norm and bring about S_4. While the violation may be detected, there will still be a reduction to organisational performance and under certain circumstances this may be severe (for example, where violation causes loss of resources or damage to the system). To reduce the risk of reaching S_4, a sub-goal obligation to reach S_2 can be added. By being obliged to reach S_2, the agent is obliged not to reach the alternative S_1, which is a pre-requisite for S_4. Of course, this can still be violated by an agent, but it would be noticed earlier and would not damage organisational performance. If monitoring is effective, the organisation could remove the agent from the role before reaching S_4 and before the damage.

One approach to determining what kind of probationary contract to use (or where to add a sub-goal obligation) is thus to iterate through the possible states in the graph adding a sub-goal obligation as above, calculate the risk for each contract and choose the lowest. This distinguishes between different probationary contracts corresponding to different probationary roles. However, the probationary contract with the lowest risk may not be the best choice, because while an organisation may wish to reduce risk, it must also distinguish between trustworthy and untrustworthy agents by means of observing their performance in probationary roles. Here, if a probationary contract removes too much risk, then performance of agents is hard to distinguish.

Indeed, by constraining agent behaviour with an additional sub-goal obli-

gation, perfectly trustworthy agents may be rejected because of their range of capability. For example, the standard role for the task in Figure 12 allows two ways to perform a task, but if an agent is only capable of performing it via S_1, then it will not be able to fulfil a probationary contract that obliges S_2. Therefore, an organisation must be mindful of the capabilities of the candidate agents when it selects a probationary contract.

Consider an abstract organisation that requires external agents to perform services and uses a *gatekeeper* to allocate these service provider roles. Each role has a single task (specified by a single contract), and the organisation has norms targeting these roles in order to improve organisational performance. For clarity, we consider only the case in which an organisation seeks to reduce the risk of an agent violating a norm to save effort or otherwise increase its personal reward at the expense of organisational performance.

In our evaluation, we assumed we have a pool of 400 agents, which are homogenous except for their probability of violation, P_{viol}, which is chosen from a uniform random distribution from 0 to 1, $P_{viol} \in \mathbb{R} | 0 \leq P_{viol} \leq 1$. To ensure some dynamism, every 100 units of time, each idle agent is removed from the pool with probability 0.002, so there will always be unknown agents in the pool. Our example organisation requires 40 agents at any one time to fill roles and perform tasks. When it has a vacancy, the organisation's gatekeeper chooses an agent from the pool at random and assesses it using a simple trust mechanism based on previous interactions to decide whether to allow an agent to play the role. After an agent completes a task, an interaction value, iv, is generated, where $iv = \frac{perceived\ performance}{possible\ performance}$. Possible performance is the maximum performance accrued from a job completed with no violations, and perceived performance is calculated by reducing the possible performance by the impact of detected norm violations (undetected violations do not count against an agent). The interaction value has a maximum value of 1.0, but the minimum is open-ended, since violating norms may result in organisational performance being reduced sufficiently to result in a negative value. The trust value is calculated as the mean of the interaction values. For an agent a the trust value is

$$trust_a = \frac{1}{|I_a|} \sum_{iv_i \in I_a} iv_i$$

where I_a is the set of previous interactions for agent a, and iv_i is one of the interactions in that set. The gatekeeper uses two trust thresholds: accept, τ_{accept}, and reject, τ_{reject}. If $trust_a \geq \tau_{accept}$ then the agent is accepted for a standard role, and if $trust_a < \tau_{reject}$ it is rejected outright. Otherwise, the agent is accepted for a probationary role. If an agent is rejected for a role, another agent is selected from the pool for assessment. We use $\tau_{accept} = \tau_{reject} = 0.9$ when not using probationary roles, and $\tau_{accept} = 0.9, \tau_{reject} = 0.8$ when using probationary roles. An unknown agent is accepted for the standard role when not using probationary roles, otherwise it is accepted for a probationary role.

Figure 13 shows the performance ratio of the probationary contract method over time. When all agents are unknown, using probationary contracts is very useful, since an organisation allowing unknown agents to work in a standard contract suffers performance losses from violated norms. Over time, the benefit of using probationary contracts is reduced since the organisation builds up enough information about each agent and retains those agents that have proven

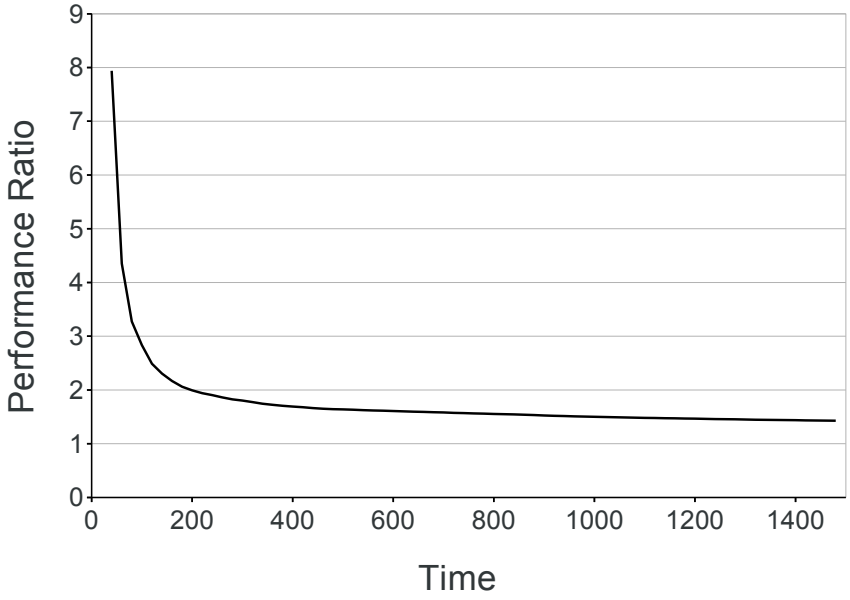

Figure 13: Performance improvement ratio using probationary contracts

to be trustworthy. However, using probationary contracts continues to have some benefit, albeit a less dramatic one.

4.6 Resource-constrained peer enforcement

In the previous section, we considered norms that are imposed and monitored by central authorities. They can be a valuable mechanism for regulating or constraining the behaviour of self-interested agents. However, in virtual environments, interactions can be of high magnitude and speed, and thus their regulation is expensive, and may even be infeasible. Social norms offer a means to provide distributed mechanisms for the self-regulation of virtual systems and societies, by delegating to the population itself the responsibility to impose appropriate behavioural standards [3, 18].

In this context, many have been concerned with the development of mechanisms to ensure the emergence of such social norms. In particular, researchers from many scientific areas have considered *punishment* as a key motivating element for norms to be established [7, 10, 13, 20]. Here, punishment is a utility-based incentive, typically incurring an enforcement cost for the punisher, but bringing a potential benefit to the population as a whole when correctly applied. Work that investigates the use of punishment as a means for social norms to emerge has assumed that agents applying such punishment have unlimited resources, allowing them to bear the resulting enforcement cost. This assumption is significant in real world settings in which resources are limited and require more careful exploitation. For example, sensors in wireless networks have limited energy and thus need to optimise their use of it.

Peer punishment has been widely used as a method to regulate the behaviour

of participants in a distributed system. Such punishment usually carries a cost for the agent applying the punishment, known as an enforcement cost. Most existing models that study this phenomenon make use of *static* punishment, where there is an enforcement cost paid by the enforcer that is fixed at design time. Considering the free riding phenomenon in a peer-to-peer (P2P) file sharing system, de Pinninck et al. [5] suggest that the punishment for such behaviour should be *blocking*, by which all other agents cease interacting with an agent that is observed not to share files after downloading them. The blocking period can be for a specific length of time, which needs to be set at design time. In this case, the enforcement cost paid is the loss of access to files from the blocked agents.

Determining an appropriate blocking period can be crucial to the performance of the overall system, especially those that rely on the participation of members for their functionality and effectiveness (as with P2P networks). Because of the dynamism of these systems and the cost incurred by both the agent that is applying the punishment and the agent that is being punished, choosing a single punishment value that is effective to be used against all agents may not be feasible. Therefore, mechanisms that support punishments whose value can be adapted at run-time are much better suited. This is *adaptive* punishment. Returning to our example, a blocking time of 30 minutes may be enough to deter the behaviour of agent i, but not agent j. The next time that agent j defects, a blocking time that is longer than 30 minutes can be used to try to force agent j to comply and start sharing files.

In Axelrod's metanorm model [1], a population of agents play a game in which each agent has to decide between cooperation and defection. The agent population evolves through a number of iterations, with a mechanism whereby successful behaviour (as measured by the scoring system) tends to be replicated and unsuccessful behaviour tends to be discarded. Axelrod's model is limited due to the evolutionary approach taken, meaning that agents are removed from the system and new ones created which is often not feasible in a practical running system. Later work replaced the evolution with a reinforcement learning algorithm that limits accessibility to global information, and instead allows agents to learn from their own experience [15]. Moreover, in order to capture a key feature of computational systems such as on-line virtual communities, Axelrod's classic model has been adapted by introducing a topological structure [14] that determines observability among agents, so that an agent's neighbours are the only witnesses of its interactions. This indicates that an agent only imposes punishment on its defecting neighbours, and metapunishment on its non-punishing neighbours.

In the metanorm model, agents play a game iteratively. In each iteration, each agent must decide between cooperation and defection. Defection brings a reward for the defecting agent called a *temptation* value, and a penalty to all other agents called a *hurt value*. However, each defector risks being observed *by the other agents* in the population, and punished as a result. These other agents thus decide whether to punish agents that were observed defecting, with a low penalty for the punisher known as the *enforcement cost* and a high penalty for the punished agent known as the *punishment cost*. Agents that do not punish those observed defecting risk being observed themselves, and potentially incur metapunishment. Thus, finally, each agent decides whether to metapunish agents observed to spare defecting agents. Again, metapunishment comes at a

high penalty for the punished agent and a low penalty for the punisher, through the punishment cost and enforcement cost, respectively.

The decisions of agents are driven by two private variables: *boldness*, and *vengefulness*. Boldness determines the probability that an agent defects, and vengefulness is the probability that an agent punishes or metapunishes another agent. In each round, agents are given a fixed number of opportunities to defect, in which boldness determines the probability that an agent defects, and vengefulness is the probability that an agent punishes or metapunishes another agent. Thus, the boldness and vengefulness of an agent are said to comprise that agent's *policies*. After several rounds of the game, each agent's rewards and penalties are tallied, and successful and unsuccessful strategies are identified. Having performed a set of actions in a particular round, agents are able to adapt their policies according to the positive or negative outcomes of these policies using a policy learning algorithm.

We first evaluated the effects of limiting resources on the outcome of the metanorm model with static punishment. Axelrod originally suggested the proportion of (-9,-2) between punishment cost and enforcement cost, and our work is consistent with this. The interaction model of agents involves the following sequence of actions. For every defection opportunity that an agent i has, all of i's neighbours have a chance to punish i. If a neighbour j decides to spare i from punishment, then all of j's neighbours have the chance of metapunishing j. Assuming that every agent has 4 distinct neighbours, this means that for every single defection, 4 punishment decisions need to be taken, and if all these punishment decisions result in sparing the defector, $4 \times 4 = 16$ metapunishments can arise. Based on this, it is clear that agents will invest most of their resources on punishment and metapunishment as a result of the outcome of the first few defections, with scarce resources left to regulate the behaviour of the remaining agents.

The results showed that the model fails to establish the norm with the average boldness of agents remaining very high, and reflecting a very high rate of defection. The surprise here is that the average vengefulness is also high. Previous reported results of the metanorm model have shown that high vengefulness and high boldness is not a stable state for the population and usually leads to a norm establishment state with high vengefulness and low boldness. The case here is different because of the introduction of the limited resources. High vengefulness is due to two factors. First, for the first few occurrences of defection, sufficient resources remain available for metapunishment. Second, resources run out quickly, so no more enforcement costs are being paid by agents to cause vengefulness to drop. This last factor can also be used to explain the high boldness, with insufficient punishment taking place to deter defecting agents by outweighing the temptation gained.

While static punishment does not always establish the norm under limited resources settings, it seems that adaptive punishment should do better, since the punishment is adapted according to the *image* of the agent under punishment or metapunishment. Thus resources should be used efficiently. The results shown in Figure 14 confirm this. In this experiment the punishment was factorised based on a basic punishment unit of -1, and the cost of punishment (enforcement cost) is set to 1 unit for punishers, reducing the utility of violators by 4 units (1:4 proportion is used because it has been shown [19] to be more effective in promoting cooperation). The results are better since the level of boldness

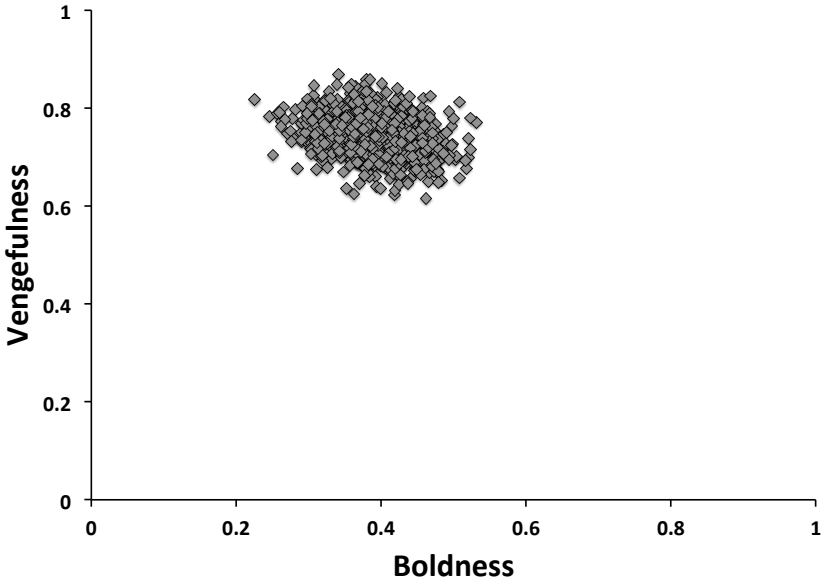

Figure 14: Impact of limited resources with adaptive punishment on final boldness and vengefulness

drops to a mid-range level. However, this does not reflect norm establishment, since a considerable number of defections still take place in every round. This is because adaptive punishment allows better use of available resources to regulate the behaviour of other agents, but resources are still not being made best use of, since the adaptive punishment mechanism in its current form depends only on the image of the current defector.

We introduced a new version of the adaptive punishment technique that is capable of allocating an appropriate amount of resources for the current enforcement action, taking into account the possibilities of future violations and the resources needed to deal with such violations [16].

The model of adaptive punishment used is that each agent is supplied with a memory of limited size, in which they store information about their observations of actions taken by their direct neighbours. So, in the case of observing a defection, an agent stores the identity of the defecting agent together with the fact that this agent has defected, and the same in the case of cooperation. An agent that spares a defector from punishment is also a defector, while one that punishes a defector is a cooperator. These facts are also stored in the memory, together with the identity of agents.

In order to allow agents to apply punishment with the appropriate intensity, punishment needs to change according to the defector's previous history. Based on this, and in relation to a particular *defecting* agent j, two main factors can be calculated: the number of previous instances of defection of agent j (dentoted by nd_j), and the number of previous instances of compliance of agent j (dentoted by nc_j), both in the context of the window size. From these values we obtain the *defection proportion* (dentoted by dp_j), representing the percentage of defections

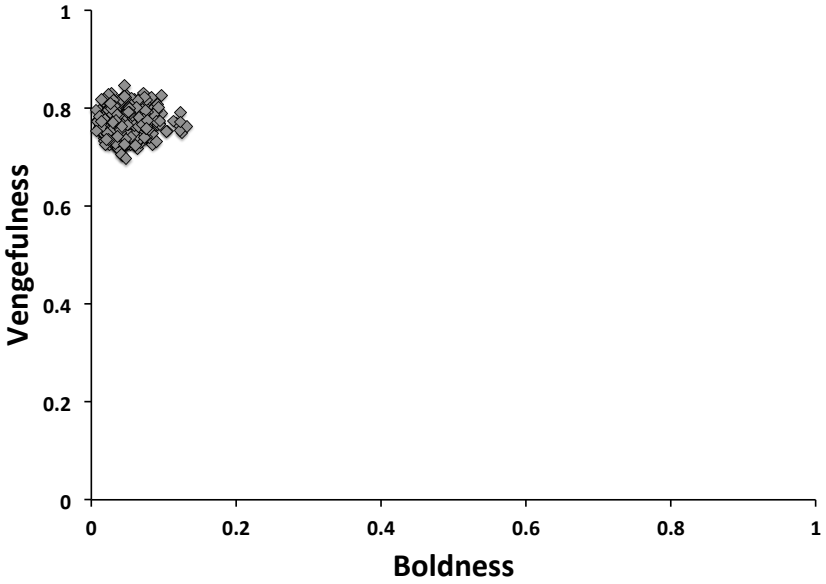

Figure 15: Impact of limited resources with resource-aware adaptive punishment

compared to the total number of decisions, by dividing nd_j by the total of nd_j and nc_j.

The value of the adaptive punishment in this case is a factor of the behaviour of the defecting agent with regard to defection, in comparison to the rest in the neighbourhood, the uniform resources available to deal with agents on an equal basis and an initial punishment unit. This means that if the agent at hand has a defection rate that is larger than average, then the uniform resources are scaled up to deal with this agent, and in the case that the agent defection rate is less than average, then the uniform resources are scaled down. The value of meta-punishment is calculated similarly, with the number of defections representing the number of instances of sparing defectors, and the number of instances of compliance representing the number of instances of punishing defectors.

The results of experiments with our new model are shown in Figure 15, which indicate considerable improvements from the previous results, as norm establishment has been observed in all runs. Even when the neighbourhood size is increased, norm establishment is achieved. This is due to the capability of the new approach to consider the number of neighbours that an agent may have to deal with, so punishment may be less even with more neighbours. However, the increased number of neighbours compensates for this.

5 Conclusions

We believe that the Engineering Emergence in Large-Scale Simulations (EELS) project has generated interesting insights and valuable new approaches to controlling complex systems in which behaviour emerges from autonomous inter-

acting components. A thorough review of existing literature identified gaps in current research. This informed the work of the EELS project and a journal paper was published from this review [11].

A formal model of emergent norms has been developed, extending an event-based formalism of emergent behaviour. This model allows automated reasoning about emergent norms and will act as a foundation to subsequent work on norm identification, evaluation and commitment. A norm detection mechanism has been developed based upon our event based formalism. It uses a multidimensional concept of emergent norms in order to refine norm hypotheses. We have developed an explicit representation of expectations and integrated them into an agent reasoning cycle.

We have designed and implemented multiple mechanisms for controlling the emergence and adoption of beneficial norms under different system circumstances: a mutual norm commitment protocol, whereby a group of agents can agree to commit to a newly emerged norm in order to provide firm expectations of future behaviour; a probationary contract mechanism restricting agents' permissions while we explore their trustworthiness; and, a mechanism for punishing violation of beneficial norms under resource-constrained conditions. The latter two results have been published in conference papers [12, 16].

Looking to the future, EELS focused on detecting behaviour as it emerges and engineering systems to take advantage of where this is positive. To have more control over a complex system, we need to predict where and what new behaviours may emerge. To do that, we first need to explain why behaviour has emerged (previously in reality or in simulations of the future), so that we know the basis on which to predict.

References

[1] R. Axelrod. An evolutionary approach to norms. *The American Political Science Review*, 80(4):1095–1111, 1986.

[2] C. Bicchieri. *The Grammar of Society: The nature and dynamics of social norms*. Cambridge University Press, 2006.

[3] R. Boyd, H. Gintis, S. Bowles, and P. J. Richerson. The evolution of altruistic punishment. *Proceedings of the National Academy of Sciences of the United States of America*, 100(6):3531–3535, 2003.

[4] C. Castelfranchi, F. Giardini, E. Lorini, and L. Tummolini. The prescriptive destiny of predictive attitudes: From expectations to norms via conventions. In *Proceedings of the 25th Annual Meeting of the Cognitive Science Society*, pages 222–227, 2003.

[5] A. P. de Pinninck, C. Sierra, and M. Schorlemmer. Distributed Norm Enforcement: Ostracism in Open MultiAgent Systems. In *Computable Models of the Law*, volume 4884 of *LNCS*, pages 275–290. Springer, 2008.

[6] M. d'Inverno, M. Luck, P. Noriega, J. Rodriguez-Aguilar, and C. Sierra. Communicating open systems. *Artificial Intelligence*, 186:138–94, 2012.

[7] E Fehr and S. Gächter. Altruistic punishment in humans. *Nature*, 415(6868):137–140, January 2002.

[8] M. Finnemore and K. Sikkink. International norm dynamics and political change. *International organization*, 52(04):887–917, 1998.

[9] P. Gärdenfors. The role of expectations in reasoning. In *In Michael Masuch and Laszlo Polos, editors, Knowledge Representation and Reasoning under Uncertainty, LNCS 808*, 1994.

[10] F. Giardini, G. Andrighetto, and R. Conte. A cognitive model of punishment. In *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*, pages 1282–1288. Austin: Cognitive Science Society, 2010.

[11] C. Haynes, M. Luck, P. McBurney, S. Mahmoud, T. Vítek, and S. Miles. Engineering the emergence of norms: a review. *The Knowledge Engineering Review*, 32, 2018.

[12] C. Haynes, S. Miles, and M. Luck. Probationary contracts: Reducing risk in norm-based systems. In M. Rovatsos, G. Vouros, and V. Julian, editors, *Multi-Agent Systems and Agreement Technologies: 13th European Conference, EUMAS 2015, and Third International Conference, AT 2015, Athens, Greece, December 17-18, 2015, Revised Selected Papers*, volume 9571 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2016.

[13] D. Helbing, A. Szolnoki, M. Perc, and G. Szabas. Punish, but not too hard: how costly punishment spreads in the spatial public goods game. *New Journal of Physics*, 12(8):083005, 2010.

[14] S. Mahmoud, J. Keppens, M. Luck, and N. Griffiths. Norm establishment via metanorms in network topologies. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, WI-IAT '11, pages 25–28. IEEE Computer Society, 2011.

[15] S. Mahmoud, J. Keppens, M. Luck, and N. Griffiths. Overcoming omniscience in axelrod's model. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Volume 03*, WI-IAT '11, pages 29–32. IEEE Computer Society, 2011.

[16] S. Mahmoud, S. Miles, and M. Luck. Cooperation emergence under resource-constrained peer punishment. In *Proceedings of the Fifteenth International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[17] S. Modgil, N. Oren, N. Faci, F. Meneguzzi, S. Miles, and M. Luck. Monitoring compliance with e-contracts and norms. *Artificial Intelligence and Law*, 23(2):161–196, 2015.

[18] N. Nikiforakis. Punishment and counter-punishment in public good games: Can we really govern ourselves? *Journal of Public Economics*, 92:91–112, 2008.

[19] N. Nikiforakis and H. Normann. A comparative statics analysis of punishment in public-good experiments. *Experimental Economics*, 11(4):358–369, 2008.

[20] D. Villatoro, G. Andrighetto, J. Sabater-Mir, and R. Conte. Dynamic sanctioning for robust and cost-efficient norm compliance. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 414–419. IJCAI/AAAI, 2011.

Algorithm 1 Algorithm to detect norms from events

Require: Event store, E
Require: Domain knowledge DK
Require: Refinement policy, R
Require: Significance threshold, ϵ_{sig}
Require: Extent threshold, ϵ_{ext}
Require: Salience threshold, ϵ_{sal}

Generate CES, $CES \leftarrow getCES(DK, E)$
$refine \leftarrow true$
while $refine = true$ **do**
  $M \leftarrow match(E, CES)$
  $causal \leftarrow checkCausality(M, E, DK)$
  if $causal = false$ **then**
    $CES \leftarrow refine(CES, R)$
    continue
  end if
  $significance \leftarrow checkSig(M, E, DK)$
  if $significance < \epsilon_{sig}$ **then**
    $CES \leftarrow refine(CES, R)$
    continue
  end if
  $inevitable \leftarrow checkInev(M, E, DK)$
  if $inevitable = true$ **then**
    $CES \leftarrow refine(CES, R)$
    continue
  end if
  $extent, salience \leftarrow getExtSal(M, E, DK)$
  if $extent > \epsilon_{ext}$ AND $salience > \epsilon_{sal}$ **then**
    $norm \leftarrow getNorm(CES, DK)$
    $refine = false$
  else
    $CES \leftarrow refine(CES, R)$
  end if
end while
return hypothetical norm, $norm$