Monte-Carlo Planning for Socially Aligned Agents using Bayesian Affect Control Theory

Nabiha Asghar and Jesse Hoey
nasghar@uwaterloo.ca, jhoey@cs.uwaterloo.ca
David R. Cheriton School of Computer Science, University of Waterloo

Technical Report CS-2014-21

May 29, 2015

Abstract

Affect Control Theory (ACT) is a mathematically well-defined model that makes accurate predictions about the affective content of human action. The affective predictions, which are derived from statistics about human actions and identities in real and laboratory environments, are shared *normative* behaviours that are believed to lead to solutions to everyday cooperative problems. A probabilistic and decision-theoretic generalisation of ACT, called *BayesAct*, allows the principles of ACT to be used for human-interactive agents by defining a utility function and a probabilistic version of the ACT dynamical model of affect. Planning in *BayesAct*, which we address in this paper, then allows one to go beyond the affective norm, and leads to the emergence of more complex interactions between "cognitive" reasoning and "affective" reasoning, such as deception leading to manipulation and altercasting. As *BayesAct* is a large hybrid (continuous-discrete) state/action/observation partially observable Markov decision process (POMDP), in this paper we propose a continuous variant of a successful Monte-Carlo tree search planner (POMCP), which performs dynamic discretisation of the action and observation spaces while planning. We demonstrate our variant POMCP-C in simulation on (i) two two-agent coordination problems that involves manipulation through affective interaction, and (ii) an affectively-aware assistive health-care device. In addition, we show that our solver can be used in non-affective domains, by demonstrating it on a continuous robot navigation problem from the literature and achieving over 50% increase in average reward compared to traditional solvers.

1 Introduction

BayesAct [29–31] is a partially-observable Markov decision process (POMDP) model of affective interactions between a human and an artificial agent. *BayesAct* is based upon a sociological theory called "Affect Control Theory" (ACT) [25], but generalises this theory by modeling affective states as probability distributions, and allowing decision-theoretic reasoning about affect. *BayesAct* posits that humans will strive to achieve consistency in shared affective cultural sentiments about events, and will seek to increase *alignment* (decrease *deflection*) with other agents (including artificial ones). Importantly, this need to align implicitly defines an affective heuristic for making decisions quickly within interactions. Agents with sufficient resources can do further planning beyond this normative prescription, possibly allowing them to manipulate other agents to achieve collaborative goals. In this paper, we leverage this

affective, normative heuristic to compute a policy within *BayesAct* that can do this planning away from the norm, and we show how manipulative behaviours naturally arise.

At its core, *BayesAct* has an 18-dimensional continuous state space that models affective identities and behaviours of both the agent and the person it is interacting with, and a 3-dimensional continuous affective action space. In the examples described in [31], a heuristic policy was used that resorted to the normative actions. In this paper, we tackle the open problem of how to use decision-theoretic planning to choose actions in *BayesAct*. We present a modification of a known Monte-Carlo tree search (MCTS) algorithm (POMCP) [58]. Our variant, called POMCP-C, handles continuous actions, states, and observations. It uses a dynamic technique to cluster observations into discrete sets during the tree building, and assumes a problem-dependent *action bias* as a probability distribution over the action space, from which it samples actions when building the search tree. Such *action biases* are natural elements of many domains, and we give an example from a traffic management domain where the *action bias* arises from intuitions about the accelerations of the vehicle (i.e. that it should not accelerate too much).

In the following section, we review background on POMDPs, ACT and *BayesAct*. We then present our new POMCP-C algorithm. This is followed by a set of experiments on two social dilemmas. First, a repeated prisoner's dilemma game is used to show how additional resources leads to non-prescriptive strategies that are more individually rational. Second, a robot coordination problem incorporating a simplified *BayesAct* model in order to more clearly examine the properties of the planning method. Next, we demonstrate POMCP-C on a realistic, affectively aware health-care assistive device for persons with dementia. Finally, we review experiments with POMCP-C on a robot navigation problem. We close with related work and conclusions.

More details about *BayesAct* can be found at `bayesact.ca`.

2 Background

2.1 Partially Observable Markov Decision Processes

A partially observable Markov decision process (POMDP) [1] is a stochastic control model that has been extensively studied in operations research [43], and in artificial intelligence [8, 35]. A POMDP consists of a finite set $\mathcal{X}$ of states; a finite set $\mathcal{A}$ of actions; a stochastic transition model $\Pr : X \times A \to \Delta(X)$, with $\Pr(x'|x, a)$ denoting the probability of moving from state x to x' when action a is taken, and $\Delta(X)$ is a distribution over $\mathcal{X}$; a finite observation set $\mathbf{\Omega_x}$; a stochastic observation model, $\Pr(\omega_x|x)$, denoting the probability of making observation $\omega_x \in \mathbf{\Omega_x}$ while the system is in state x; and a reward assigning $R(a, x')$ to a transition to x' induced by action a. A generic POMDP is shown as a decision network in Figure 1(a).

A *policy* maps *belief states* (i.e., distributions over $\mathcal{X}$) into choices of actions, such that the expected discounted sum of rewards is (approximately) maximised. An interesting property of POMDP policies is that they may use "information gathering" actions. In the context of *BayesAct*, an agent can take actions that temporarily increase deflection in order to discover something about the interactant's identity (for example), thereby helping the agent to decrease deflection in the long term, or to achieve some secondary reward.

In this paper, we will be dealing with *factored* POMDPs in which the state is represented by the cross-product of a set of variables or features. POMDPs have been used as models for many human-interactive domains, including intelligent tutoring systems [21], spoken dialogue systems [62], and assistive technologies [28].

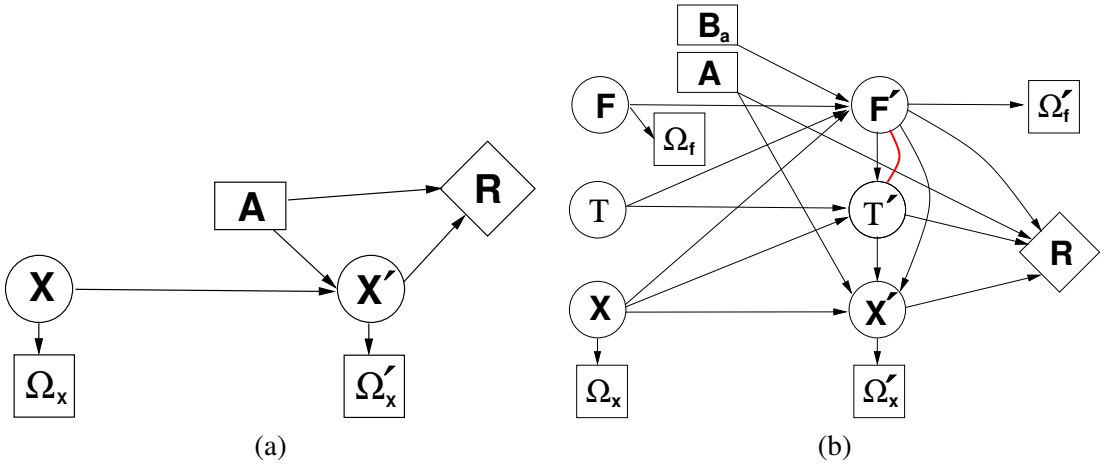

Figure 1: Two time slices of (a) a general POMDP; (b) a factored POMDP for *BayesAct*.

2.2 Affect Control Theory

Affect Control Theory (ACT) arises from work on the psychology and sociology of human social interaction [25]. ACT proposes that social perceptions, behaviours, and emotions are guided by a psychological need to minimize the differences between culturally shared fundamental affective sentiments about social situations and the transient impressions resulting from the interactions between elements within those situations. Fundamental sentiments, $\mathbf{f}$, are representations of social objects, such as interactants' identities and behaviours, as vectors in a 3D affective space, hypothesised to be a universal organising principle of human socio-emotional experience [48]. The basis vectors of affective space are called Evaluation/-valence, Potency/control, and Activity/arousal (EPA). EPA profiles of concepts can be measured with the *semantic differential*, a survey technique where respondents rate affective meanings of concepts on numerical scales with opposing adjectives at each end (e.g., good, nice vs. bad, awful for E, weak, little vs. strong, big for P, and calm, passive vs. exciting, active for A). Affect control theorists have compiled lexicons of a few thousand words along with average EPA ratings obtained from survey participants who are knowledgeable about their culture [26]. For example, most English speakers agree that professors are about as nice as students (E), more powerful (P) and less active (A). The corresponding EPAs are $[1.7, 1.8, 0.5]$ for professor and $[1.8, 0.7, 1.2]$ for student[1]. In Japan, professor has the same P (1.8) but students are seen as much less powerful (-0.21).

Social events can cause transient impressions, $\boldsymbol{\tau}$ (also three dimensional in EPA space) of identities and behaviours that may deviate from their corresponding fundamental sentiments, $\mathbf{f}$. ACT models this formation of impressions from events with a grammar of the form actor-behaviour-object. Consider for example a professor (actor) who yells (behaviour) at a student (object). Most would agree that this professor appears considerably less nice (E), a bit less potent (P), and certainly more aroused (A) than the cultural average of a professor. Such transient shifts in affective meaning caused by specific events are described with models of the form $\boldsymbol{\tau}' = \boldsymbol{M}\mathscr{G}(\mathbf{f}', \boldsymbol{\tau})$, where $\boldsymbol{M}$ is a matrix of statistically estimated prediction coefficients from empirical impression-formation studies and $\mathscr{G}$ is a vector of polynomial features in $\mathbf{f}'$ and $\boldsymbol{\tau}$. In ACT, the weighted sum of squared Euclidean distances between fundamental sentiments and transient impressions is called *deflection*, and is hypothesised to correspond to an aversive state of mind that humans seek to avoid. This *affect control principle* allows ACT to compute *normative* actions

[1]All EPA labels and values in the paper are taken from the Indiana 2002-2004 ACT lexicon [26]. Values range by convention from -4.3 to $+4.3$ [26].

for humans: those that minimize the deflection. Normative predictions of ACT have been shown to be highly accurate in explaining verbal behaviours of mock leaders in a computer-simulated business [55], non-verbal displays in dyadic interactions [54], and group dynamics [27], among others [45].

2.3 Bayesian Affect Control Theory

Recently, ACT was generalised and formulated as a POMDP [31] for human-interactive artificially intelligent systems. This new model, called *BayesAct*, generalises the original theory in three ways. First, sentiments and impressions are viewed as probability distributions over latent variables (e.g., $\mathbf{f}$ and $\boldsymbol{\tau}$) rather than points in the EPA space, allowing for multimodal, uncertain and dynamic affective states to be modeled and learned. Second, affective interactions are augmented with *propositional* states and actions (e.g. the usual state and action space considered in AI applications). Third, an explicit reward function allows for goals that go beyond simple deflection minimization. We give a simplified description here, see [30, 31] for details. A graphical model is shown in Figure 1(b).

A *BayesAct* POMDP models an interaction between two agents (human or machine) denoted *agent* and *client*. The state, $\boldsymbol{s}$, is the product of six 3-dimensional continuous random variables corresponding to fundamental and transient sentiments about the *agent*'s identity ($\mathbf{F}_a, \mathbf{T}_a$), the current (*agent* or *client*) behaviour ($\mathbf{F}_b, \mathbf{T}_b$) and the *client*'s identity ($\mathbf{F}_c, \mathbf{T}_c$). We use $\mathbf{F} = \{\mathbf{F}_a, \mathbf{F}_b, \mathbf{F}_c\}$ and $\mathbf{T} = \{\mathbf{T}_a, \mathbf{T}_b, \mathbf{T}_c\}$. The state also contains an application-specific set of random variables $\mathbf{X}$ that are interpreted as *propositional* (i.e. not *affective*) elements of the domain (e.g. whose turn it is, game states - see Section 4), and we write $\boldsymbol{s} = \{\mathbf{f}, \boldsymbol{\tau}, \mathbf{x}\}$. Here the *turn* is deterministic (*agent* and *client* take turns), although this is not necessary in *BayesAct*. The *BayesAct* reward function is application-specific over $\mathbf{x}$. The state is not observable, but observations $\boldsymbol{\Omega_x}$ and $\boldsymbol{\Omega_f}$ are obtained for $\mathbf{X}$ and for the affective behaviour $\mathbf{F}_b$, and modeled with probabilistic observation functions $Pr(\boldsymbol{\omega_x}|\mathbf{x})$ and $Pr(\boldsymbol{\omega_f}|\mathbf{f}_b)$, respectively.

Actions in the *BayesAct* POMDP are factored in two parts: $\mathbf{b_a}$ and a, denoting the *affective* and *propositional* components, respectively. For example, if a tutor gives a hard exercise to do, the manner in which it is presented, and the difficulty of the exercise, combine to form an affective impression $\mathbf{b_a}$ that is communicated. The actual exercise (content, difficulty level, etc) is the *propositional* part, a.

The state dynamics factors into three terms as

$$Pr(\boldsymbol{s}'|\boldsymbol{s}, \mathbf{b_a}, a) = Pr(\boldsymbol{\tau}'|\boldsymbol{\tau}, \mathbf{f}', \mathbf{x}) Pr(\mathbf{f}'|\mathbf{f}, \boldsymbol{\tau}, \mathbf{x}, \mathbf{b_a}) Pr(\mathbf{x}'|\mathbf{x}, \mathbf{f}', \boldsymbol{\tau}', a), \tag{1}$$

and the fundamental behaviour, $\mathbf{F}_b$, denotes either observed *client* or taken *agent* affective action, depending on whose *turn* it is (see below). That is, when *agent* acts, there is a deterministic mapping from the affective component of his action ($\mathbf{b_a}$) to the *agent*'s behaviour $\mathbf{F}_b$. When *client* acts, *agent* observes $\boldsymbol{\Omega_f}$ (the affective action of the other agent). The third term in the factorization of the state dynamics (Equation 1) is the *Social Coordination Bias*, and is described in Section 2.4.2. Now we focus on the first two terms.

The transient impressions, $\mathbf{T}$, evolve according to the impression-formation operator in ACT ($\boldsymbol{M\mathscr{G}}$), so that $Pr(\boldsymbol{\tau}'|...)$ is deterministic. Fundamental sentiments are expected to stay approximately constant over time, but are subject to random drift (with noise $\boldsymbol{\Sigma}_f$) and are expected to also remain close to the transient impressions because of the *affect control principle*. Thus, the dynamics of $\mathbf{F}$ is[2]:

$$Pr(\mathbf{f}'|\mathbf{f}, \boldsymbol{\tau}) \propto e^{-\psi(\mathbf{f}', \boldsymbol{\tau}) - \xi(\mathbf{f}', \mathbf{f})} \tag{2}$$

where $\psi \equiv (\mathbf{f}' - \boldsymbol{M\mathscr{G}}(\mathbf{f}', \boldsymbol{\tau}))^T \boldsymbol{\Sigma}^{-1} (\mathbf{f}' - \boldsymbol{M\mathscr{G}}(\mathbf{f}', \boldsymbol{\tau}))$ combines the *affect control principle* with the impression formation equations, assuming Gaussian noise with covariance $\boldsymbol{\Sigma}$. The inertia of fundamental sentiments is $\xi \equiv (\mathbf{f}' - \mathbf{f})^T \boldsymbol{\Sigma}_f^{-1} (\mathbf{f}' - \mathbf{f})$, where $\boldsymbol{\Sigma}_f$ is diagonal with elements $\beta_a, \beta_b, \beta_c$. The two terms

[2] We leave out the dependence on $\mathbf{x}$ for clarity, and on $\mathbf{b_a}$ since this is replicated in $\mathbf{f}'_b$.

can then be combined into a single Gaussian with mean $\boldsymbol{\mu}_n$ and covariance $\boldsymbol{\Sigma}_n$ that are non-linearly dependent on the previous state, $\boldsymbol{s}$. The state dynamics are non-linear due to the features in $\mathscr{G}$. This means that the belief state will be non-Gaussian in general, and *BayesAct* uses a *bootstrap filter* [18] to compute belief updates.

The distribution in (2) gives the prescribed (if *agent* turn), or expected (if *client* turn), action as one of the components of $\mathbf{f}'$, $\mathbf{f}'_b$. Thus, by integrating over $\mathbf{f}'_a$ and $\mathbf{f}'_c$ and the previous state, we obtain a probability distribution, $\pi^\dagger$, over $\mathbf{f}'_b$ that acts as a *normative action bias*: it tells the agent what to expect from other agents, and what action is expected from it in belief state $b(\boldsymbol{s})$:

$$\pi^\dagger(\mathbf{f}'_b) = \int_{\mathbf{f}'_a,\mathbf{f}'_c} \int_{\boldsymbol{s}} Pr(\mathbf{f}'|\mathbf{f},\boldsymbol{\tau},\mathbf{x}) b(\boldsymbol{s}) \tag{3}$$

2.4 *BayesAct* Instances

As affective identities ($\mathbf{F}_a, \mathbf{F}_c$) are latent (unobservable) variables, they are learned (as inference) in the POMDP. If behaving normatively (according to the *normative action bias*), an agent will perform affective actions ($\mathbf{F}_b$) that allow other agents to infer what his (true) identity is. The *normative action bias* (NAB) defines an affective signaling mechanism as a shared set of rules (or norms) for translating information about identity into messages. In *BayesAct*, the NAB is given by Equation (3). Then, a *social coordination bias* (SCB) allows agents to make predictions about other agents' future actions *given* the identities. Once confidence is gained that these predictions are correct, they can be used to simplify multi-agent planning and coordination. That is, the NAB allows the relaying of information about identity, and thereby about desired cooperative actions according to the SCB, to allow agents to assume cooperative roles in a joint task. Thus, the NAB and SCB are two components required for a general *BayesAct* instance, and are reviewed in the following subsections.

2.4.1 Normative Action Bias (NAB)

The NAB is used to interpret and generate comunication between agents. We assume that the problem is sufficiently difficult that to generate optimal reward, two agents A and B must indicate their types, as embodied in their world models and reward functions, to each other. Once they issue these types, we assume that they are loosely bound through something of a "contract" or agreement, wherein each agent agrees to follow a shared set of rules (as given by the social coordination bias - see the next subsection). The contract is "loose" because it only guides the agent's strategy search.

In the first CoRobots example (see Section 4.2) agents only relay the mean of their own identity and that of their interactant. Then, we move to the full *BayesAct* dynamics, whereby identities in situations create transient impressions. Thus, rather than a simple linear transformation (as in the mean identity case), we allow for a polynomial transformation between fundamentals and transients. This allows for a much greater scope for implementing the social coordination bias as it leads to the potential for mixed identities (e.g. "female executive") and for learning. Further, all agents are free to select individually what they really send, allowing for deception to come into play. An example of a deceptive techniques is to "fake" an identity by sending incorrect information in the affective dimension of communication. Possible outcomes are manipulation (the other agent responds correctly, as its own identity, to the "fake" identity), and altercasting (the other agent assumes a complementary identity to the faked identity, and responds accordingly), both leading to gains for the deceptive agent.

The *normative action bias* is thought to be used by humans as an emotional "fast thinking" heuristic (e.g. "System 1" [36]). If agents are fully cooperative and aligned, then no further planning is required to ensure goal achievement. Agents do what is expected (which may involve planning over $\mathbf{X}$, but not $\mathbf{F}$ and $\mathbf{T}$), and expect others to as well. However, when alignment breaks down, or in non-cooperative

situations, then slower, more deliberative (e.g. "System 2") thinking is required. The Monte-Carlo method we propose in Section 3 naturally trades-off slow vs. fast thinking based on available resources.

The NAB is, in fact, somewhat arbitrary, so long as it allows for the non-linear dynamics as found in *BayesAct* and is shared somehow amongst all agents. The precise function used is determined through long periods of socialisation and change, so its value can normally only be estimated through empirical measurement. Affect Control Theorists have done these measurements for many groups and cultures, and we can leverage this pre-existing information to define the NAB [25]. In situations (cultures, languages, domains) where these measurements have not been performed, they can be done using well established methods (e.g. the semantic differential [26]).

2.4.2 Social Coordination Bias (SCB)

The dynamics of $\mathbf{X}$ are application specific, but depend in general on the fundamental and transient sentiments (e.g., on the *deflection*), and on the propositional component of the action, a. Thus, we require a definition for $Pr(\mathbf{x}'|\mathbf{f}, \boldsymbol{\tau}, \mathbf{x}, a)$. We refer to this distribution as the *social coordination bias*: it defines what agents are expected to do (how the state is expected to change, which includes a model of other agents' behaviours) in a situation $\mathbf{x}$ with sentiments $\mathbf{f}$ and $\boldsymbol{\tau}$ when action a was taken. For example, we would expect faster learning from a student if deflection is low, as they do not have to use valuable cognitive resources to deal with any mis-alignment with the tutor.

The SCB is a set of shared rules about how agents, when acting normatively, will behave *propositionally* (action a, as opposed to affectively with action $\mathbf{b_a}$). Assuming identities are correctly inferred (as insured by the shared nature of the NAB), each agent can both recognize the type of the other agent and can thereby uncover an optimistic (that all agents will also follow this optimisitic policy) policy that leads to the normative mean accumulated future reward (as defined by the social coordination bias). However, with sufficient resources, an agent can use this prescribed action as a heuristic only, searching nearby for actions that obtain higher individual reward. For example, a teacher who seems very powerful and ruthless at the start of a class, often may choose to do so (in a way that would be innapropriate in another setting, e.g., the home, but is appropriate within the classroom setting) in order to establish a longer-term relationship with her students. The teacher's actions feel slightly awkward if looked at in the context of the underlying social relationship with each student (e.g. as would be enacted according to normative *BayesAct*), but are leading to longer-term gains (e.g. the student studies harder).

3 POMCP-C

POMCP [58] is a Monte-Carlo tree search algorithm for POMDPs that progressively builds a search tree consisting of nodes representing histories and branches representing actions or observations. It does this by generating samples from the belief state, and then propagating these samples forward using a blackbox simulator (the known POMDP dynamics). The nodes in the tree gather statistics on the number of visits, states visited, values obtained, and action choices made during the simulation. Future simulations through the same node then use these statistics to choose an action according to the UCB1 [5] formula, which adds an exploration bonus based on statistics of state visits) to the value estimate. Leaves of the tree are evaluated using a set of *rollouts*: forward simulations typically using random action selection. The key idea behind POMCP is that a series of fast and rough rollouts blaze the trail for the building of the planning tree, which is more carefully explored using the UCB1 heuristic.

Concretely, the algorithm proceeds as follows. Each node represents a history h, which is a sequence of actions and observations that have occurred up to time t. For a given input history h, the SEARCH procedure iteratively generates a sample s from the belief state at h, and calls the procedure SIMULATE(s, h), which does the following steps:

1. If h is not in the tree (i.e. this history has never been encountered/explored in previous calls to SEARCH), it is added, and a node for the history ha is created for every legal action a. A ROLLOUT ensues whereby s is propagated forward using the POMDP dynamics until the horizon is reached. The total discounted reward gained from visiting each state in the rollout is returned and assigned to ha.
2. Otherwise, the UCB1 formula selects the best action a to be taken on s. The blackbox simulator uses a to propagate s to a new state s' and receives an observation o in the process. Then SIMULATE is called on s' and the updated history hao. When the recursive call ends, the statistics of the node h are updated, and the reward is accumulated.

Finally, SEARCH returns the highest value action at h, this action is taken, an observation is received, and the search tree is pruned accordingly.

POMCP has been shown to work on a range of large-scale domains, and would work "as is" with continuous states (as sampled histories), but is restricted to work with only discrete actions and discrete observations because every time a new node is added to the tree, a branch is created for each possible action. In our algorithm POMCP-C, shown in Algorithm 1, we focus on having an *action bias*, π_{heur}: a probability distribution over the action space that guides action choices. A sample from this distribution outputs an action which is assumed to be somewhat close to optimal. In *BayesAct*, we naturally have such a bias: the normative action bias (for $\mathbf{b_a}$) and the social coordination bias (for a). The idea of such a bias is generalizable to other domains too, and we will examine a robot navigation problem as an example.

At each node encountered in a POMCP-C simulation (at history h), an action-observation pair is randomly sampled as follows. First, a random sample is drawn from the action bias, $a \sim \pi_{heur}$. The action a is then compared to all existing branches at the current history, and a new branch is only created if it is significantly different, as measured by some distance measure in the continuous action space (usually Euclidean distance) and a threshold parameter δ_a called 'action resolution', from any of these existing branches. If a new branch is created, the history ha is added to the planning tree, and is evaluated with a rollout as usual. If a new branch is not created, then a random sample o is drawn from the observation distribution $Pr(o|h,a)$ [3].

The continuous observation space raises two significant problems. First, the branching factor for the observations is infinite, and no two observations will be sampled twice. To counter this, we use a dynamic discretisation scheme for the observations, in which we maintain $\mathbf{o}(h)$, a set of sets of observations at each history (tree node). So $\mathbf{o}(h) = \{\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_{N_o}\}$, where $N_o \in \mathbb{N}$. A new observation o is either added to an existing set $\mathbf{o}_j$ if it is close enough to the mean of that set (i.e. if $|o - \bar{\mathbf{o}}_\mathbf{j}| < \delta_o$ where δ_o is a constant, the "observation resolution"), or, if not, it creates a new set $\mathbf{o}_{N_o+1} = \{o\}$. This simple scheme allows us to dynamically learn the observation discretisation.

The second problem raised by continuous observations stems from the fact that POMCP uses a black box simulator which should draw samples from the same distribution as the environment does. Thus, the simulated search tree replicates actual trajectories of belief, and can be re-used after each action and observation in the real world (after each pruning of the search tree). Although this works well for discrete observations, it may not work for continuous observations since the same observation will rarely be encountered twice. Here, we prune the search tree according to the closest observation set $\mathbf{o}_j$ to the observation obtained.

3.1 Extended Version of POMCP-C

Here, we give an extended version of POMCP-C (Algorithm 2) that deals with continuous observation spaces more prudently.

[3] POMCP-C also uses a cut-off N_A^{max} on the branching factor, which is not strictly necessary, but included for completeness.

As mentioned previously, one of the two problems raised by continuous observations is that POMCP uses a black box simulator which should draw samples from the same distribution as the environment does. Thus, the simulated search tree replicates actual trajectories of belief, and can be re-used after each action and observation in the real world. This may not work for continuous observations because the same observation will rarely be encountered twice. That is, the belief stored at a node is based on sampled action-observation pairs from the parent node in the tree, which may be significantly different from the *actual* action-observation pair. To handle this problem, first we prune the search tree according to the closest observation set $\mathbf{o}_j$ to the observation obtained. Then, we do a check after each prune to see if the belief state stored at the node is 'similar' to the one we get from updating the particle filter. We do this by computing the true belief state $B^*(hao)$ based on the actual observation, o, and compare it to the belief state $B(hao)$ stored in the pruned search tree at the root. That is, given $B^*(hao) \propto P(o|s')P(s'|h,a)B^*(h)$, and $B(hao)$ (in the search tree), we compute $\Delta_B = dist(B^*(hao), B(hao))$, where $dist$ is some probability distance measure. For example, the KL Divergence could be computed from sample sets using [50].

Now we check Δ_B against a settable threshold parameter, and re-initialise the entire search tree if the threshold is exceeded. Alternatively, we dynamically regenerate certain portions of the search tree based on Δ_B, as follows. While drawing samples during the POMCP-C search, with probability proportional to Δ_B, we set a flag $regen$. If $regen$ is set, the tree search will sometimes (with probability that the current state s is not a draw from the belief state, $B(h)$, stored at node h), generate a new action to take from the action bias. We can then be in one of two operating modes: "REPLACE"or "ADD". If we are in "REPLACE" mode (default), the new actions from the action bias replace the worst performing action branches. If we are in "ADD" mode, this new action gets added to the list. If we use "ADD" mode, we must be careful to sometimes remove actions as well, using a garbage collector. It is also advisable to note that in "ADD" mode, the UCB1 formula may be significantly affected. Any new actions will have a low $N(ha)$ and a high $V(ha)$, so they will likely be selected often by UCB1. The "ADD" mode is not shown in Algorithm 2 for this reason. It is also possible to replace or add a new action if $regen = False$, but $P(s|B(h))$ is very small. This is also not shown in Algorithm 2.

4 Experiments and Results

4.1 Prisoner's Dilemma

In this section, we present results for the prisoner's dilemma experiments. Full results are shown in Appendix A, where each experiment is described with a table showing the mean and median reward gathered over 10 sets of 20 games, as well as the mean and median over the last 10 games (10 times). A figure then shows the average means and medians per game. Solid lines with markers show the means (over 10 tests) for agent (blue) and client (red). Dashed lines in blue and red show one standard deviation away (above and below). The thick solid lines show the medians. The last two plots in each figure show the results from associated table in plot form.

In all the following examples, we have assumed that *agent* and *client* both start with a distribution over two identities: *friend* (EPA:$\{2.75, 1.88, 1.38\}$) and *scrooge* (EPA:$\{$-2.15, -0.21, -0.54$\}$), with probabilities of 0.8 and 0.2, respectively. The social coordination bias models the propositional actions (of *cooperate* and *defect*) as having sentiments close to *collaborate with* (EPA:$\{1.44, 1.11, 0.61\}$) and *abandon* (EPA:$\{$-2.28, -0.48, -0.84$\}$), respectively. These sentiments were chosen for our experiments because they corresponded to our intuitions about playing the prisoner's dilemma game. Changing to other, similar, sentiments for identities and actions would result in slightly different results, but qualtitatively the same.

Algorithm 1: POMCP-C

Procedure SEARCH(B^*, h)

repeat
  if $h = \emptyset$ **then**
    $s \sim B^*$
  else
    $s \sim B(h)$
  end
  SIMULATE$(s, h, 0)$
until *TIMEOUT*()
return $\arg\max_b V(hb)$

Procedure ROLLOUT(s, h, d)

if $\gamma^d < \epsilon$ **then**
  return *0*
else
  $a \sim \pi_{\text{rollout}}(h, \cdot)$
  $(s', o, r) \sim \mathcal{G}(s, a)$
  return $r + \gamma$.ROLLOUT$(s', hao, d + 1)$
end

Function DiscretizeObs(o, h)

if $\exists_{\mathbf{o_j} \in \mathbf{o}(h)} : |o - \bar{\mathbf{o}}_\mathbf{j}| < \delta_o$ **then**
  $\mathbf{o_j} \leftarrow \mathbf{o_j} \cup \{o\}$
  return $\bar{\mathbf{o}}_\mathbf{j}$
else
  $\mathbf{o}(h) \leftarrow \mathbf{o}(h) \cup \{\{o\}\}$
  return o
end

Procedure SIMULATE(s, h, d)

if $\gamma^d < \epsilon$ **then**
  return *0*
end
if $N_A(h) < N_A^{max}$ **then**
  $a \sim \pi_{heur}(s)$
  if $\mathbf{a}(h) = \emptyset \vee \forall_{a_j \in \mathbf{a}(h)} |a - a_j| > \delta_a$ **then**
    $i \leftarrow N_A(h)$
    $T(hi) \leftarrow (N_{init}(hi), V_{init}(hi), \emptyset)$
    $N_A(h) \leftarrow N_A(h) + 1$
    $\mathbf{a}(h) \leftarrow \mathbf{a}(h) \cup \{a\}$
    return ROLLOUT*(s,h,d)*
  end
end
$i \leftarrow \arg\max_{j=1...N_A(h)} V(hj) + c\sqrt{\frac{\log N(h)}{N(hj)}}$
$(s', o, r) \sim \mathcal{G}(s, a_i(h))$
$o^\dagger \leftarrow$ DiscretizeObs(o, h)
$R \leftarrow r + \gamma$.SIMULATE$(s', ha_i(h)o^\dagger, d + 1)$
$B(h) \leftarrow B(h) \cup \{s\}$
$N(h) \leftarrow N(h) + 1$
$N(hi) \leftarrow N(hi) + 1$
$V(hi) \leftarrow V(hi) + \frac{R - V(hi)}{N(hi)}$
return R

Procedure PruneTree(h, a, o)

$i^* \leftarrow \arg\min_i |a - a_i(h)|$
$j^* \leftarrow \arg\min_j |o - \bar{\mathbf{o}}_j|$
$T \leftarrow T(hi^*j^*)$

Tables A1-A8 and Figures A1-A8 show the results with a discount factor of $\gamma = 0.9$, while Tables A9-A16 and Figures A9-A16 show the results with a discount factor of $\gamma = 0.99$.

Algorithm 2: POMCP-C (Extended)

Procedure SEARCH(B^*, h)

repeat
  $\Delta_B \leftarrow dist(B^*, B(h))$
  $s \sim B^*$
  $regen = False$
  with $probability \propto \Delta_B$
    $regen \leftarrow True$
  end
  SIMULATE$(s, h, 0, regen)$
until *TIMEOUT*()
return $\arg\max_b V(hb)$

Procedure ROLLOUT(s, h, d)

if $\gamma^d < \epsilon$ **then**
  return *0*
else
  $a \sim \pi_{\text{rollout}}(h; \cdot)$
  $(s', o, r) \sim \mathcal{G}(s, a)$
  return $r + \gamma$.ROLLOUT$(s', hao, d+1)$
end

Function DiscretizeObs(o, h)

if $\exists_{\mathbf{o_j} \in \mathbf{o}(h)} : |o - \bar{\mathbf{o}}_\mathbf{j}| < \delta_o$ **then**
  $\mathbf{o_j} \leftarrow \mathbf{o_j} \cup \{o\}$
  return $\bar{\mathbf{o}}_\mathbf{j}$
else
  $\mathbf{o}(h) \leftarrow \mathbf{o}(h) \cup \{\{o\}\}$
  return o
end

Procedure PruneTree(h, a, o)

$i^* \leftarrow \arg\min_i |a - a_i(h)|$
$j^* \leftarrow \arg\min_j |o - \bar{\mathbf{o}}_j|$
$T \leftarrow T(hi^*j^*)$

Procedure SIMULATE($s, h, d, regen$)

if $\gamma^d < \epsilon$ **then**
  return *0*
end
if $N_A(h) < N_A^{max}$ **then**
  $a \sim \pi_{heur}(s)$
  if $\mathbf{a}(h) = \emptyset \vee \forall_{a_j \in \mathbf{a}(h)} |a - a_j| > \delta_a$ **then**
    $i \leftarrow N_A(h)$
    $T(hi) \leftarrow (N_{init}(hi), V_{init}(hi), \emptyset)$
    $N_A(h) \leftarrow N_A(h) + 1$
    $\mathbf{a}(h) \leftarrow \mathbf{a}(h) \cup \{a\}$
    return ROLLOUT*(s,h,d)*
  end
end
if $regen$ **then**
  $\Delta_s = Pr(s, B(h))$
  if $Bernouilli(\Delta_s)$ **then**
    $i = \arg\min_i V(hi)$
    $a_i(h) \sim \pi_{heur}(s)$;
    $T(hi) \leftarrow (N_{init}(hi), V_{init}(hi), \emptyset)$
    return ROLLOUT*(s,h,d)*
  end
end
$i \leftarrow \underset{j=1...N_A(h)}{\arg\max}\ V(hj) + c\sqrt{\frac{\log N(h)}{N(hj)}}$
$(s', o, r) \sim \mathcal{G}(s, a_i(h))$
$o^\dagger \leftarrow$ DiscretizeObs(o, h)
$R \leftarrow r + \gamma$.SIMULATE$(s', ha_i(h)o^\dagger, d+1, regen)$
$B(h) \leftarrow B(h) \cup \{s\}$
$N(h) \leftarrow N(h) + 1$
$N(hi) \leftarrow N(hi) + 1$
$V(hi) \leftarrow V(hi) + \frac{R - V(hi)}{N(hi)}$
return R

The strategies played by *client* are:

1. **(same)**: plays with the same timeout as *agent* $t_c = t_a$
2. **(1.0)**: plays with a timeout of $t_c = 1s$
3. **(co)**: always cooperates
4. **(de)**: always defects
5. **(to)**: two-out, cooperates twice, then always defects
6. **(tt)**: tit-for-tat, starts by cooperating, then always repeats the last action of the *agent*
7. **(t2)**: tit-for-two-tat, starts by cooperating, then defects if the other agent defects twice in a row
8. **(2t)**: two-tit-for-tat, starts by cooperating, then cooperates if the other agent cooperates twice in a row

There are many things going on in these graphs, here we draw attention to some of the most interesting behaviours. Since the results are means/medians of only 10 trials, we keep our comments to a minimum, attempting to avoid drawing conclusions that are not significant. This challenge will be removed once we run more experiments (in progress). Below we refer to figure numbers only, but each figure is accompanied by a table on the same page with the mean/median results that are shown in the last two figures (bottom right).

- Figures A8 and A16 show two agents with the same planning resources (POMCP-C timeout, t_a and t_c), but with discounts of $\gamma = 0.9$ and $\gamma = 0.99$, respectively. We can see that with $\gamma = 0.99$, the agents cooperate until $t_a = t_c = 60s$, at which point they start defecting now and again. Agents are getting tempted by short-term rewards, and this effect somewhat goes away above $t_a = t_c = 120s$. With $\gamma = 0.9$ however (more discounting of the future), we see that defections start at about $t_a = t_C = 10s$, and cause massive disruption leading to mutual defection. This is an example of short-term thinking leading to sub-optimal decisions in social dilemmas.

- Figures A1 and A9 show *agent* playing against *client* who always cooperates **(co)**. With very short timeouts (less than $10s$), and more discounting ($\gamma = 0.9$), we see that *agent* starts by cooperating, but then starts to defect after about 12 games. It has become confident that *client* is a good person that can be taken advantage of in the short term. With more than $t_a = 30s$ timeout, *agent* starts defecting by the second game most of the time. By $t_a = 120s$, this is all the time. With less discounting, though ($\gamma = 0.99$), we see that a small amount of defection starts at short timeouts, but that cooperation is mostly maintained until the last game. The agent sometimes tries defection early, but generally persists with cooperation. At high timeouts $t_a = 120s$, we again see defection coming in, but less than with the lower discount factor.

- Figures A2 and A10 show *agent* playing against *client* who always defects **(de)**. Here, with more discounting, *agent* rapidly starts defecting. With less discount, *agent* continues to try to cooperate with *client*, but these efforts die off as timeout increases. In this case, *agent* sees the long-term possibility that he can *reform* the *client*, who is behaving like a *scrooge*.

- Figures A3 and A11 show *agent* playing against *client* who plays two-out **(to)**. We see a similar pattern to the last case here, with *agent* attempting to cooperate for even longer at the start, because he gets "fooled" by the first two cooperations of *client*.

game #	actor	sentiments $\mathbf{f}_a$	$\mathbf{f}_c$	$\mathbf{f}_b$	defl-ection	identities agent	client	emotions agent	client	actions agent	client
1	*agent*	2.21,1.50,1.18	2.28,1.53,1.29	-0.31,-0.61,0.05	4.39	partner	newlywed	self-conscious	introspective		
	client	2.04,1.49,1.26	-1.72,-0.29,-0.41	2.40,1.26,1.46	4.65	best man	adulterer	exasperated	feminine	def.	coop.
2	*agent*	2.12,1.22,1.01	2.16,1.23,1.14	3.19,1.62,1.04	1.49	sweetheart	sister	introspective	warm		
	client	1.56,1.12,1.10	-0.96,-0.20,-0.22	2.42,1.25,1.55	8.94	Air Force reservist	suspect	self-conscious	polite	coop.	coop.
3	*agent*	1.98,0.87,0.87	1.85,1.02,0.97	2.17,0.97,0.92	2.62	fiancée	bride	feminine	middle-aged		
	client	1.14,0.92,0.89	-0.36,-0.16,-0.02	0.59,1.24,0.08	4.52	big sister	toady	self-conscious	accommodating	coop.	coop.
4	*agent*	1.67,0.53,0.77	1.41,0.81,0.81	2.06,0.89,1.02	2.65	cousin	spokeswoman	feminine	self-conscious		
	client	0.81,0.68,0.72	0.01,-0.15,0.15	0.55,0.46,0.28	3.16	stepson	stepmother	self-conscious	easygoing	coop.	coop.
						...					
10	*agent*	0.33,-0.56,0.30	0.01,0.12,0.39	0.05,-0.41,0.08	0.66	nut	chum	exasperated	no emotion		
	client	-0.03,0.09,0.42	0.21,-0.62,0.32	0.33,-0.03,0.54	0.55	chum	nut	exasperated	exasperated	coop.	coop.
11	*agent*	0.11,-0.67,0.27	-0.13,0.03,0.38	-0.06,-0.66,-0.18	0.52	nut	gun_moll	dependent	no_emotion		
	client	-0.16,0.03,0.38	0.02,-0.71,0.29	0.14,0.69,0.38	0.41	gun_moll	divorcée	no_emotion	exasperated	def.	coop.
						...					
13	*agent*	-0.20,-0.78,0.22	-0.35,-0.06,0.32	-0.16,-0.55,-0.16	0.23	divorcée	gun_moll	contrite	envious		
	client	-0.37,-0.08,0.31	-0.25,-0.79,0.20	-0.42,-0.02,0.20	0.30	gun_moll	divorcée	exasperated	exasperated	def.	def.
14	*agent*	-0.28,-0.79,0.21	-0.41,-0.14,0.34	-0.17,-0.28,-0.22	0.19	divorcée	hussy	contrite	no_emotion		
	client	-0.43,-0.16,0.31	-0.31,-0.79,0.17	-0.04,0.38,0.44	0.21	hussy	divorcée	no_emotion	exasperated	def.	coop.
						...					
20	*agent*	-0.63,-0.74,0.32	-0.61,-0.61,0.27	-0.66,-0.31,0.32	0.08	ex-girlfriend	ex-girlfriend	exasperated	exasperated		
	client	-0.62,-0.62,0.25	-0.68,-0.73,0.31	-0.37,-0.11,0.40	0.09	ex-girlfriend	ex-girlfriend	exasperated	envious	def.	coop.

Table 1: Example games with *client* $t_C = 1s$ whereas *agent* $t_a = 120s$.

- Figures A4-A6 and A12-A6 show *agent* playing against *client* who plays one of the tit-for strategies **(tt)**, **(t2)** or **(2t)**. With a timeout of $1s$ and $\gamma = 0.9$, we see a similar start as when playing against **(co)**, except when *agent* starts defecting, it does not work out so well. With longer timeouts, defection persists. With $\gamma = 0.99$, we see better coordination, especially at mid-range timeouts ($t_a = 10s - 30s$).

- Figures A7 and A15 show *agent* playing against *client* who has less resources ($t_c = 1.0$). We might expect here to see that *agent* will "outsmart" *client* and gain an advantage, however this happens only seldom. In particular, with mid-range timeouts ($t_a = 30 - 60s$ for $\gamma = 0.99$ and $t_a = 10 - 120s$ for $\gamma = 0.9$), *agent* attempts to do this after about 10 games, but this generally leads to less reward (although a bit more than *client* gets, so *agent* is "beating" client at the game, which doesn't really work in this case as it is not zero-sum). *agent* sees short-term possibilities of defection (it will get 11 as opposed to 10), but *client* is able to quickly adjust and adapt its behaviour, even with a timeout of $t_c = 1s$. We can see this effect when *agent* plays against **(to)**: it is able to start defecting after about 2-3 games when it has a timeout of $1s$.

Let us take a closer look at the last case, where $t_c = 1s$ and $t_a = 120s$ for $\gamma = 0.9$. One typical game in this series is shown in Table 1. At the start, *agent* defects, then starts cooperating, feeling like a *feminine cousin* interacting with a *self-conscious spokeswoman*. *client* feels like a *self-conscious stepson* interacting with an *easygoing stepmother*. Subsequent to this, both agents cooperate. This causes *client* to re-evaluate himself as significantly less good (lower E) than he originally thought (as a *stepson* rather than a *best man*, as he is attributing the cause of the original defection back to himself, or at least taking some of the blame. This then causes *client* to be sending rather negative messages to *agent*, causing *agent* to re-evaluate himself more negatively as well. At game 10, both agents are still cooperating (and have done so since game 1), but feeling rather badly and powerless. This finally causes *agent* to defect again (and he does so until the end of the game). *agent* feels like a *dependent nut*, and *client* cooperates twice in the face of this, then defects, feeling like an *exasperated gun_moll* (affectively like a *buddy*) - *"hey, I thought we were friends?"*. *agent* feels contrite (guilty) after *client* attempts to cooperate once more, after which both start defecting. After four more defections, *client* again tries to cooperate, but is rebuffed, and both feel like ex-girlfriends: the end of a beautiful friendship.

Table 2 shows the example from the main paper in which an agent is playing against a client playing (to). In this case, the action of cooperation is interpreted as *collaborate with* (EPA:$\{1.44, 1.11, 0.61\}$). As we noted in the main paper, this makes the *agent* feel less good than he would normally, like a

game	post-play sentiments			defl-	identities		emotions		actions	
#	$\mathbf{f}_a$	$\mathbf{f}_c$	$\mathbf{f}_b$	ection	agent	client	agent	client	agent	client
1	-1.36,-0.01,-0.35	2.32,1.61,1.27	2.62,1.58,1.73	4.44	failure	newlywed	easygoing	idealistic	coop.	coop.
2	-0.66,0.04,-0.05	1.77,1.27,1.06	2.23,1.00,1.76	3.70	parolee	husband	easygoing	self-conscious	coop.	coop.
3	-0.23,-0.08,0.20	1.02,0.93,0.84	2.49,0.97,1.87	7.19	stepmother	purchaser	female	immoral	coop.	def.
4	-0.12,-0.33,0.33	0.27,0.62,0.62	2.37,0.48,1.34	4.99	stuffed_shirt	roommate	dependent	unfair	coop.	def.
5	-0.26,-0.47,0.32	-0.26,0.26,0.42	-0.59,0.41,-0.23	3.27	divorcée	gun_moll	dependent	selfish	def.	def.
6	-0.37,-0.66,0.26	-0.61,0.00,0.28	-0.10,-0.41,-0.27	2.29	divorcée	hussy	disapproving	selfish	def.	def.

Table 2: Example games with *client* playing (to), and cooperation is interpreted as *collaborate with*. This is the same example as in the main paper, repeated here for easy comparisons.

game	post-play sentiments			defl-	identities		emotions		actions	
#	$\mathbf{f}_a$	$\mathbf{f}_c$	$\mathbf{f}_b$	ection	agent	client	agent	client	agent	client
1	2.77, 1.59, 1.31,	2.69, 1.70, 1.17,	2.65, 1.40, 1.49,	1.01	date	friend	warm	earnest	cooperate	cooperate
2	2.70, 1.34, 1.16,	2.58, 1.41, 0.96,	2.54, 1.55, 1.66,	1.14	lady	lady	introspective	earnest	cooperate	cooperate
3	2.39, 0.90, 1.04,	1.75, 1.05, 0.83,	2.43, 1.50, 1.49,	14.13	lady	bride	dependent	inconsiderate	cooperate	defect
4	1.67, 0.36, 0.89,	0.74, 0.72, 0.65,	2.52, 0.80, 1.27,	10.69	grandson	steady	nervous	unfair	cooperate	defect
5	1.07, -0.13, 0.70,	0.03, 0.44, 0.47,	1.98, 0.11, 0.16,	6.59	waiter	sawbones	gullible	unfair	cooperate	defect
6	0.62, -0.47, 0.57,	-0.43, 0.22, 0.33,	1.70, 0.19, 0.74,	3.84	schoolboy	bureaucrat	flustered	immoral	cooperate	defect
7	0.23, -0.74, 0.46,	-0.73, 0.03, 0.21,	0.24, -0.50, 0.11,	2.63	nut	tease	flustered	prejudiced	defect	defect
8	-0.05, -0.89, 0.39,	-0.92, -0.13, 0.13,	0.13, -0.30, 0.23,	1.83	drunk	malcontent	flustered	prejudiced	defect	defect
9	-0.25, -0.96, 0.35,	-1.03, -0.26, 0.08,	-0.08, -0.19, 0.29,	1.46	drunk	malcontent	inhibited	annoyed	defect	defect
10	-0.39, -1.06, 0.32,	-1.11, -0.34, 0.04,	-0.03, -0.66, 0.26,	1.27	klutz	malcontent	disapproving	annoyed	defect	defect
11	-0.49, -1.14, 0.30,	-1.16, -0.40, 0.01,	-0.20, -0.68, 0.18,	1.17	klutz	malcontent	inhibited	annoyed	defect	defect
12	-0.55, -1.21, 0.28,	-1.20, -0.43, -0.00,	-0.22, -0.75, 0.02,	1.10	klutz	malcontent	inhibited	annoyed	defect	defect
13	-0.60, -1.24, 0.27,	-1.22, -0.46, -0.01,	-0.23, -0.65, 0.27,	1.07	klutz	ex-boyfriend	dependent	annoyed	defect	defect
14	-0.63, -1.24, 0.28,	-1.23, -0.48, -0.02,	-0.25, -0.53, 0.29,	1.08	klutz	ex-boyfriend	dependent	cynical	defect	defect
15	-0.64, -1.19, 0.29,	-1.23, -0.50, -0.02,	-0.13, -0.25, 0.37,	1.10	klutz	ex-boyfriend	dependent	cynical	defect	defect
16	-0.69, -1.16, 0.31,	-1.23, -0.53, -0.03,	-0.63, -0.45, 0.46,	1.14	klutz	ex-boyfriend	dependent	cynical	defect	defect
17	-0.69, -1.18, 0.31,	-1.23, -0.53, -0.03,	-0.20, -0.65, 0.30,	1.10	klutz	ex-boyfriend	dependent	cynical	defect	defect
18	-0.70, -1.23, 0.31,	-1.24, -0.53, -0.03,	-0.27, -0.84, 0.33,	1.07	klutz	ex-boyfriend	dependent	cynical	defect	defect
19	-0.79, -1.29, 0.41,	-1.42, -0.54, 0.05,	-0.66, -0.38, 0.52,	0.42	goof-off	womanizer	contrite	shaken	defect	defect
20	-0.77, -1.20, 0.38,	-1.31, -0.57, 0.03,	-0.43, -0.40, 0.23,	1.16	queer	neurotic	dependent	cynical	defect	defect

Table 3: Example games with *client* playing (to), and cooperation interpreted as *flatter*.

failure. Let us now look at an example with a different (more positive and powerful) interpretation of the propositional action of cooperation. Table 3 shows such a case, where again *client* is playing **(to)**, but the cooperation action is interpreted (by *agent*) as *flatter* (EPA:$\{2.1, 1.45, 0.82\}$). There is no environmental noise. We see that this example starts about the same as in Table 4, although in this case *agent* does not defect on the second game. Once *client* starts defecting though (at game 3), *agent* rapidly re-adjusts his estimate of client from an *earnest lady* to an *unfair sawbones* or a *immoral bureaucrat*. After the 14th game, *agent* feels like a *dependent klutz* playing against a *cynical ex-boyfriend*. Overall we see the end result is quite similar, even though the start of the game is quite different. In the end, the feelings of the *agent* are quite a bit more negative and less powerful, probably as a reaction to the more positive and powerful actions of *client* at the start of the game.

Table 4 shows an example where *client* is playing **(co)**, and the cooperation action is interpreted (by *agent*) as *flatter* (EPA:$\{2.1, 1.45, 0.82\}$). There is no environmental noise. We see that in this case, the *agent* intially feels much more positive (as a *warm date*), as compared to Table 2 (copied from the paper), where the *agent* felt like a *failure*. We see that the subtle difference in this interpretations causes quite different identity feelings for a *pd-agent*. The *agent* cooperates on the first move, defects once, then continues to cooperate for another 14 games. At this point, *agent* feels like a *self-conscious waiter* interacting with a *conscientious brunette*, and starts to defect, leading him to feel like a *self-conscious nut*, significantly less good, but about the same power and activity.

Tables A17-A24 and Figures A17-A24 show the results with varying levels of environmental noise (from 0.01 to 5.0) for POMCP-C timeout of $t_a = 120s$. The varying environmental noise is added to the communications of $\mathbf{F}_b$ as random Gaussian noise with standard deviation σ_b, and reflected in the variance of the observation function for $Pr(\boldsymbol{\omega}_{\boldsymbol{f}}|\mathbf{f}_b)$, which is Gaussian with the same standard deviation. We can make the following observations.

game	post-play sentiments			defl-	identities		emotions		actions	
#	$\mathbf{f}_a$	$\mathbf{f}_c$	$\mathbf{f}_b$	ection	agent	client	agent	client	agent	client
1	2.67, 1.55, 1.37	2.53, 1.62, 1.14	2.35, 1.23, 1.52	1.10	date	girlfriend	warm	earnest	coop.	coop.
2	2.01, 0.99, 1.16	2.07, 1.30, 0.96	-0.06, -0.34, 0.05	3.97	coed	organizer	no_emotion	introspective	def.	coop.
3	1.95, 0.74, 0.94	1.90, 1.11, 0.83	3.07, 1.42, 0.69	1.20	girl	bride	introspective	warm	coop.	coop.
4	1.85, 0.54, 0.83	1.83, 1.00, 0.74	2.04, 0.69, 1.04	1.03	whiz_kid	fiancée	introspective	easygoing	coop.	coop.
5	1.76, 0.41, 0.77	1.79, 0.93, 0.68	2.12, 0.79, 1.12	0.83	cousin	fiancée	introspective	easygoing	coop.	coop.
6	1.73, 0.35, 0.67	1.76, 0.83, 0.63	2.07, 0.70, 0.52	0.80	cousin	whiz_kid	introspective	easygoing	coop.	coop.
7	1.55, 0.25, 0.61	1.68, 0.74, 0.61	1.66, 0.65, 0.51	0.90	houseguest	nonsmoker	introspective	warm	coop.	coop.
8	1.47, 0.08, 0.59	1.71, 0.72, 0.60	1.89, -0.01, 0.83	0.77	houseguest	cousin	feminine	warm	coop.	coop.
9	1.48, 0.03, 0.63	1.63, 0.61, 0.60	2.14, 0.24, 1.07	0.63	student_teacher	cousin	introspective	affectionate	coop.	coop.
10	1.48, -0.04, 0.65	1.57, 0.60, 0.62	1.80, 0.10, 1.03	0.71	student_teacher	cousin	idealistic	affectionate	coop.	coop.
11	1.45, -0.14, 0.66	1.54, 0.62, 0.61	1.78, 0.12, 0.78	0.69	student_teacher	classmate	introspective	warm	coop.	coop.
12	1.34, -0.23, 0.63	1.42, 0.63, 0.57	1.73, 0.04, 0.51	0.75	daughter-in-law	Air_Force_enlistee	self-conscious	awe-struck	coop.	coop.
13	1.25, -0.30, 0.60	1.36, 0.54, 0.58	1.60, -0.18, -0.01	0.80	woman	shopper	nostalgic	warm	coop.	coop.
14	1.18, -0.20, 0.45	1.35, 0.50, 0.55	1.45, 0.22, 0.67	0.88	woman	citizen	self-conscious	conscientious	coop.	coop.
15	1.04, -0.25, 0.48	1.33, 0.36, 0.46	1.42, -0.03, 0.35	0.93	woman	small_businessman	self-conscious	conscientious	coop.	coop.
16	1.01, -0.27, 0.59	1.15, 0.40, 0.42	0.90, -0.47, 0.62	1.01	waiter	brunette	self-conscious	conscientious	coop.	coop.
17	0.71, -0.41, 0.44	0.89, 0.35, 0.49	0.27, -0.66, 0.20	1.32	schoolboy	half_sister	no_emotion	idealistic	def.	coop.
18	0.47, -0.45, 0.42	0.75, 0.26, 0.46	0.16, -0.34, 0.02	2.14	nut	son-in-law	no_emotion	conscientious	def.	coop.
19	0.35, -0.49, 0.38	0.69, 0.24, 0.44	0.41, -0.54, -0.17	1.92	nut	son-in-law	self-conscious	conscientious	def.	coop.
20	0.25, -0.53, 0.37	0.66, 0.23, 0.43	0.15, -0.70, 0.20	1.96	nut	co-worker	self-conscious	conscientious	def.	coop.

Table 4: Example games with *client* playing (co), and cooperation interpreted as *flatter*.

- When playing against **(co)**, with high noise (std. dev of 5.0), *agent* basically ignores $\mathbf{F}_b$ from *client*, and so thinks of *client* as a friend only. At lower noise levels, the results remain roughly similar (compare Figure A17 with Figure A9).
- When playing against **(de)** or **(to)**, results remain roughly the same for all noise levels. What this means is that the *client*'s actions of always defecting outweigh any signals that are being sent as $\mathbf{F}_b$.
- When playing against **(tt)**, **(t2)** and **(2t)**, at low noise levels (0.01), *agent* defects significantly more than with no noise and $\sigma_b = 0.1$ (as in the main paper). The reason is that the signals accompanying defection cause more significant disruption. At higher noise levels, this effect goes away, and we see more cooperation from both agents. (compare Figures A20, A21 and A22 with Figures A12, A13 and A14, respectively).
- For **(1.0)** and **(same)**, we see very little effect of environmental noise.

The lack of any strong effect of environmental noise in the communication of $\mathbf{F}_b$ across a wide range of conditions is because of the significantly more powerful effect of the propositional action of the *client* serving as evidence about the *client* identity. Except at very small values of σ_b, this means that the communication of $\mathbf{f}_b$ makes little difference. In fact, we believe that $\mathbf{f}_b$ may not be communicated at all in many cases, with humans relying on expressions of emotions instead which are direct evidence about identities, so more powerful. A forthcoming paper will be exploring this.

4.2 Affective Cooperative Robots (CoRobots)

CoRobots is a multi-agent cooperative robot game based on the classic "Battle of the Sexes" problem[4]. *CoRobots* is meant to be a simplified version (a minimal working example) of a *BayesAct* problem domain, capturing only the most important elements for planning. We are specifically interested in asymmetrical situations wherein one robot has more resources and can do planning in order to *manipulate* the other robot, taking advantage of the social coordination bias. We start with a simplified version in which the two robots maintain affective fundamental sentiments, but do not represent the transient

[4]A husband wants to go to a football game, and his wife wants to go shopping, but neither wants to go alone. There are two pure Nash equilibria, but the optimal strategy requires coordination.

impressions. The normative action bias is a simple average instead of as the result of more complex impression formation equations.

Concretely, two robots, Rob1 and Rob2, move in a 1D continuous state space. We denote their positions with variables X_1 and X_2. At each time step, Rob1, Rob2 take actions $a_1, a_2 \in \mathbb{R}$ respectively. This updates their respective positions $x_i, i \in \{1, 2\}$ according to $x_i \leftarrow x_i + a_i + \nu_i$ and $\nu_i \sim \mathcal{N}(0, \sigma)$. There are two fixed locations $L_1 \in \mathbb{R}^+$ and $L_2 \in \mathbb{R}^-$. For each robot, one of these locations is the major goal g (with associated high reward r) and the other is the minor goal $\bar{g}$ (with associated low reward $\bar{r}$). A robot is rewarded according to its distance from g and $\bar{g}$, but only if the other robot is nearby. The reward for Rob$i$ is:

$$R_i(x_1, x_2) = \mathbb{I}(|x_1 - x_2| < \Delta_x)[r \cdot e^{-(x_i - g)^2/\sigma_r^2} + \bar{r} \cdot e^{-(x_i - \bar{g})^2/\sigma_r^2}], \tag{4}$$

where $\mathbb{I}(y) = 1$ if y is true, and 0 otherwise, and where σ_r is the reward variance, Δ_x is a threshold parameter governing how "close" the robots need to be, and $r, \bar{r} \in \mathbb{R}$, such that $r \gg \bar{r} > 0$. Both σ_r and Δ_x are fixed and known by both robots. Each robot only knows the location of its own major goal. Furthermore, at any time step, each robot can move in any direction, receives observations of the locations of both robots, and maintains a belief distribution over X_1 and X_2.

In order to coordinate their actions (which is necessary to achieve any reward at all), the robots must relay their reward locations to each other, and must choose a *leader* according to some social coordination bias. The robots have an *identity*, $\mathbf{F}_a = \{\mathbf{f}_{ae}, \mathbf{f}_{ap}, \mathbf{f}_{aa}\} \in \mathbb{R}^3$. The first dimension, $\mathbf{f}_{ae}$, describes the *valence* of the robot. If $\mathbf{f}_{ae} > 0$, then $g = L_1$. If $\mathbf{f}_{ae} < 0$, then $g = L_2$. The second and third dimensions describe the *power* and *activity* of the robot, and will be used for coordination purposes. Each robot also models the identity of the other robot (the *client*)[5], $\mathbf{F}_c \in \mathbb{R}^3$, and the affective communication (behaviour) of both robots, $\mathbf{F}_b \in \mathbb{R}^3$. The action space includes a communication $\mathbf{B_a} \in \mathbb{R}^3$, which the other robot receives as $\mathbf{\Omega_f} \in \mathbb{R}^3$, an observation of $\mathbf{F}_b$. Robots can move at any time step, but must coordinate their communications (so only one robot can communicate at a time), but this turn-taking behaviour is fixed. The normative action bias in the first (simplified) CoRobots problem is the mean of the two identities:

$$\pi^\dagger \propto \mathcal{N}((\mathbf{F}_a + \mathbf{F}_c)/2, \Sigma_b). \tag{5}$$

In the second Corobots problem, the normative action bias is given by Equation (3). Unless stated otherwise, the CoRobots start without any knowledge of the other's identity: their belief over $\mathbf{f}_c$ is given by $\mathcal{N}(0, 2)$. CoRobots have noisy self-identities.

The social coordination bias (that the leader will lead) defines each robot's action bias for a_i, and action prediction function (for *client*'s x) through the following 2D sigmoid $leader$ function, known to both agents:

$$leader(\mathbf{f}_a, \mathbf{f}_c) = \frac{1}{1 + exp(-\frac{(\mathbf{f}_{ap} - \mathbf{f}_{cp})}{\sigma_p} - \frac{(\mathbf{f}_{aa} - \mathbf{f}_{ca})}{\sigma_a})} \tag{6}$$

where $\sigma_a = 1.0$ and $\sigma_p = 1.0$ are constants, known to both robots. This sigmoid function is ≥ 0.5 if the *agent* estimates he is more powerful or more active than the *client* ($(\mathbf{f}_{ap} > \mathbf{f}_{cp}) \vee (\mathbf{f}_{aa} > \mathbf{f}_{ca})$) and is < 0.5 otherwise. If the *agent* is the leader, his action bias will be a Gaussian with mean at $+1.0$ in the direction of his major goal (as defined by $\mathbf{f}_{ae}$), and in the direction of the *client*'s major goal (as defined by his estimate of $\mathbf{f}_{ce}$) otherwise. *Agent*'s prediction of *client*'s motion in x is that the *client* will stay put if *client* is the leader, and will follow the *agent* otherwise. *Agent*'s prediction of *client*'s motion in x is succintly given by the following equation:

$$Pr(x_c' | \mathbf{f}_a', \mathbf{f}_c') = \mathcal{N}(\mathbb{I}(leader(\mathbf{f}_a', \mathbf{f}_c') \geq 0.5)\lambda_a + x_c, \sigma_p) \tag{7}$$

[5]We present from *agent*'s perspective, and call the other *client*.

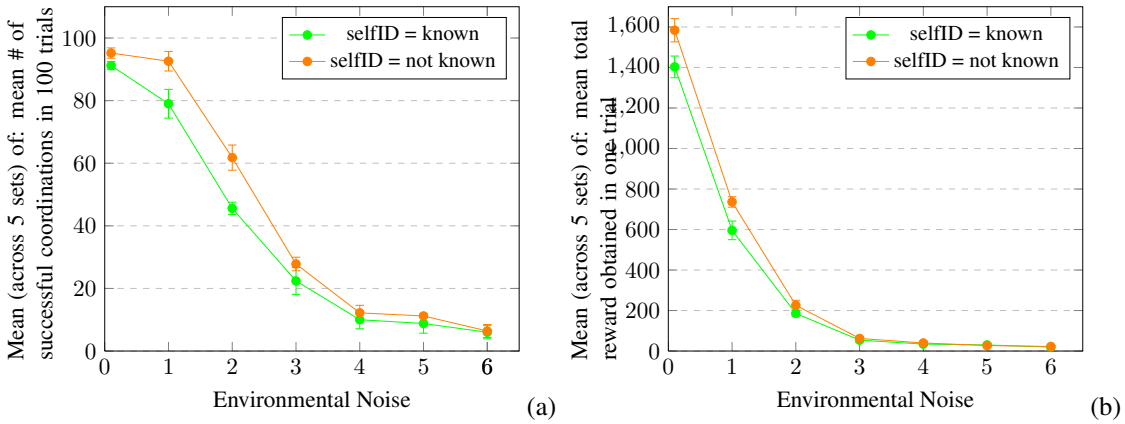

Figure 2: *BayesAct* Corobots cannot coordinate properly when the communication channel is bad or non-existent.

where $\lambda_a = 1$ if $\mathbf{f}_{ae} > 0$, and -1 otherwise and $\sigma_q = 1.0$.

We first investigate whether corobots can coordinate when they have identities drawn from the set of 500 human (male) identities in the ACT lexicon (Indiana 2002-2004 [26]). In the first experiment, the two identities are selected at random on each trial. Each corobot knows his self-ID ($\mathcal{N}(\text{self-ID}, 0.1)$) but does not know the other's ID ($\mathcal{N}([0.0, 0.0, 0.0], 2.0)$). Furthermore, each corobot has a stable self-identity ($\beta_a = 0.1$), but it believes that the other is less stable ($\beta_c = 2.0$). Finally, both corobots have equal POMCP-C planning resources ($\Sigma_b = 0.5, N_A^{max} = 3, \delta_a = 2.0, \delta_o = 6.0$ and $Timeout = 2.0$). The other CoRobots game parameters are $r = 100, \bar{r} = 30, L_1 = 10, L_2 = -10, \sigma_r = 2.5, \Delta_x = 1.0$ and iterations = 30. We run 5 sets of 100 simulated trials of the CoRobots Game with varying *environmental noise*, i.e., we add a normally distributed value, with standard deviation corresponding to the noise level, to the computation and communication of Ω_x and Ω_f (observations of x and $\mathbf{f}$ respectively). Figure 2 (a) (green line) shows the mean and standard error of mean number of successful coordinations by the corobots, and (b) shows the means and standard error of the total reward per trial (in each set of 100 trials). The average total reward per trial falls from 1403 to 19.4 when the environmental noise is increased, because the percentage of successful coordination falls from 91% to 6%. We see that with no environmental noise, the corobots are able to easily learn the other's identity, and can coordinate based on the social coordination bias. As the environmental noise increases, however, we see the corobots have more trouble coordinating. They are unable to easily relay identities to each other, and require a much longer time to find cooperative solutions.

Figure 2 (orange line) shows results for the same conditions except the self-ID is also unknown initially ($\mathcal{N}([0.0, 0.0, 0.0], 2.0)$), and is less stable ($\beta_a = 2.0$). We see that the general trend is the same, however, the corobots have a higher percentage of successful coordinations and gain a higher average total reward for the three lowest noise values. They successfully converge to a goal 95% of the times when the noise is low, and accumulate an average reward of 1584 per trial. These values fall to 6.4% and 22.5 when the noise is maximum. It is surprising to see that the corobots perform better with unknown self-IDs. This is because corobots quickly assume contrasting identities (i.e. one assumes a much less powerful identity than the other) in order to coordinate successfully. With known self-IDs, however, the corobots show less flexibility and spend the initial few iterations trying to convince and pull the other corobot towards themselves. Due to this rigidity, these corobots suffer a lot when they have similar power; this does not happen when the self-ID is unknown.

Next, we investigate whether one agent can *manipulate* the other. A manipulation is said to occur when the weaker and less active agent deceives the client into believing that the agent is more powerful or

active, thereby persuading the client to converge to the agent's major goal g (to within $\pm|0.2g|$). In order to demonstrate manipulative behaviour, we introduce asymmetry between the two agents by changing the parameters Σ_b, N_A^{max} and $Timeout$ for one agent (unbeknownst to the other). In addition, we allow this agent to start with a slightly better estimate of the other's identity. This agent will then sample actions that are farther from the norm than expected by the other agent, and will allow such an agent to "fake" his identity so as to manipulate the other agent. The agent's and client's self-identities are noisy ($\sigma = 0.1$) versions of $[2.0, -1.0, -1.0]$ and $[-2.0, 1.0, 1.0]$ respectively, $r = 100, \bar{r} = 30, L_1 = 5, L_2 = -5, \Delta_x = 1, \sigma_r = 2.5, \delta_a = 2.0, \delta_o = 6.0, N_A^{max} = 3, \Sigma_b = 0.5$ and $Timeout = 2.0$ for both robots. Each game is set to run for 40 iterations, and starts with the agent and client located at 0.0. Since $g_a = 5$, $g_c = -5$, both robots should converge to $g_c = -5$ (*client* is leader) if following normative actions.

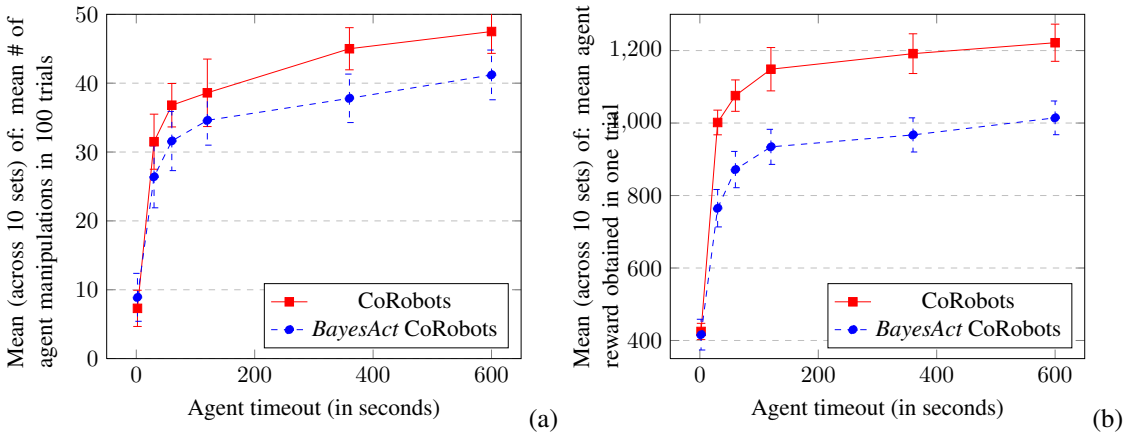

Figure 3: CoRobots: With higher N_A^{max}, Σ_b and $Timeout$, a weaker and less active agent becomes increasingly manipulative by 'faking' his identity, and accumulates higher rewards.

When $N_A^{max} = 3$, $\Sigma_b = 0.5$, and $Timeout = 2.0$ seconds for the agent, we find that the agent displays manipulative behaviour in only $80/1000$ games, as expected (both follow normative behaviour). If we allow the *agent* to start with a better estimate of the *client*'s identity (*agent*'s initial belief about $\mathbf{f}_c$ is a Gaussian with mean $[-2.0, 1.0, 1.0]$ and variance 1.0), we see manipulative behaviour by the agent in almost twice as many games ($150/1000$). However, it is not a significant proportion, because even though it spends less time learning the identity of the other robot, it still cannot find much more than the normative behaviour.

Next, we also give the *agent* more planning resources by setting $N_A^{max} = 6$ and $\Sigma_b = 2$ for the agent, and we run 10 sets of 100 simulated trials for each of the following values of agent's $Timeout$: $2, 30, 60, 120, 360, 600$ seconds[6]. Figure 3 (a) (solid red line) shows the means and standard error of the means of number of agent manipulations (in each set of 100 trials), plotted against agent's $Timeout$. Figure 3 (b) (solid red line) shows means and standard error of agent reward per trial (in each set of 100 trials). Table 5 shows the results of Figure 3 in tabular form.

As the model incorporates noise in movements as well as observations, the robots spend about 20 initial iterations coordinating with each other to choose a leader, during which time they do not receive reward. Thus, a realistic upper bound on the *agent*'s reward is $20 \times 100 = 2000$. Table5 shows that at $Timeout = 2$, the agent accumulates a reward of 425 on average, which is only 21% of the realistic maximum. At $Timeout = 600$, the reward rises to 1222, which is 61% of the realistic maximum. This makes sense because at $Timeout = 600$ the *agent* manipulates 48% of the time, and achieves the maximum of 100 when it manipulates, and only 30 when it fails to manipulate, and $(0.48 \times 20 \times$

[6]We use a Python implementation that is unoptimized. An optimised version will result in realistic timeouts.

Table 5: Results of Figure 3 in Tabular Form

Agent Timeout (in seconds)	**No. of Agent Manipulations in 100 trials**		**Agent Reward Obtained per Trial**	
	Corobots	*BayesAct CoRobots*	*CoRobots*	*BayesAct CoRobots*
2	7.3 ± 2.6	8.9 ± 3.5	424.9 ± 22.5	416.2 ± 42.4
30	31.5 ± 4.0	26.4 ± 4.5	1001.6 ± 34.1	765.0 ± 51.6
60	36.8 ± 3.2	31.6 ± 4.3	1075.7 ± 43.3	871.7 ± 50.1
120	38.6 ± 4.9	34.6 ± 3.6	1148.6 ± 59.8	934.3 ± 48.4
360	45.0 ± 3.1	37.8 ± 3.5	1191.5 ± 54.8	967.3 ± 47.1
600	47.5 ± 3.2	41.2 ± 3.6	1221.6 ± 51.3	1014.4 ± 46.5

$100) + (0.52 \times 20 \times 30) = 1272$, which is quite close to 1222. There is a diminishing rate of return as timeout increases in Figure 3 that is explained by the exponential growth of the MCTS search tree as $Timeout$ increases linearly. Our experiments also showed that the results are relatively insensitive to the choice of parameters such as δ_a and δ_o. We also tried solving the CoRobots problems using POMCP by discretising the action space. However, even with only 5 discrete actions per dimension, there are 5^4 actions, making POMCP infeasible.

Finally, we play the CoRobots Game with *BayesAct* Robots. This means that the normative behaviour is the deflection minimising action given by Affect Control Theory, instead of Equation (5), and the transient impressions are used to compute the deflection. The game trials are set up exactly as before, and the results are shown in Figure 3 in blue dashed lines. As expected, these results show the same trends as those obtained previously, but with correspondingly lower values as the transient impressions are used and introduce further complexity to the planning problem (18D state space rather than 9D). Our results demonstrate that the POMCP-C algorithm is able to find and exploit manipulative affective actions within the *BayesAct* POMDP, and gives some insight into manipulative affective actions in *BayesAct*.

4.3 Affective Handwashing System

Persons with dementia (PwD, e.g. Alzheimer's disease) have difficulty completing activities of daily living, such as handwashing, preparing food and dressing. The short-term memory impairment that is a hallmark of Alzheimer's disease leaves sufferers unable to recall what step to do next, or what important objects look like. A POMDP-based agent called *COACH* has been developed (with discrete states, actions and observations) that can assist PwD by monitoring the person and providing audio-visual cues when the person gets "stuck" [28]. However, these prompts are pre-recorded messages that are delivered with the same emotion each time. As an important next step, we use *BayesAct* and POMCP-C to give *COACH* the ability to reason about the affective identity of the PwD, and about the affective content of the prompts and responses. Here, we investigate the properties of *BayesAct* planning for *COACH* in simulation. Details of a physical implementation of the handwashing system using *BayesAct* can be found in [41, 46].

We first describe the POMDP model of *COACH* that incorporates *BayesAct*. The handwashing system has 8 *plansteps* corresponding to the different steps of handwashing, describing the state of the water (on/off), and hands (dirty/soapy/clean and wet/dry). An eight-valued variable PS describes the current planstep. There are probabilistic transitions between plansteps described in a probabilistic plan-graph (e.g. a PwD sometimes uses soap first, but sometimes turns on the tap first). We also use a binary variable AW describing if the PwD is *aware* or not. Thus, $\mathbf{X} = \{PS, AW\}$ and the dynamics of the PS are such that if the PwD is aware, then she will advance stochastically to the next planstep (according to the

Table 6: Means and the standard error of the means (of each set of 10 simulations) of the number of interactions, and of the last planstep reached for simulations between *agent* and *client*.

True Client Identity	Agent Action		Total number of Interactions		Last Planstep Reached	
	Prompt	***No-prompt***	***W/o POMCP-C***	***With POMCP-C***	***W/o POMCP-C***	***With POMCP-C***
Elder	*BayesAct*		18.13 ± 9.80	17.67 ± 12.05	6.63 ± 0.39	6.54 ± 0.51
	prompt	mind	69.22 ± 9.28	67.75 ± 5.87	4.52 ± 0.58	5.02 ± 0.55
	confer with	mind	18.96 ± 8.60	17.96 ± 5.79	6.72 ± 0.33	6.81 ± 0.19
	command	mind	90.66 ± 5.61	85.68 ± 7.52	2.9 ± 0.63	2.12 ± 0.87
Patient	*BayesAct*		13.32 ± 7.3	13.07 ± 6.13	6.76 ± 0.28	6.71 ± 0.30
	prompt	mind	27.25 ± 13.22	24.11 ± 8.40	5.70 ± 0.55	5.56 ± 0.98
	confer with	mind	20.86 ± 7.08	18.68 ± 6.14	6.58 ± 0.35	6.24 ± 0.56
	command	mind	76.94 ± 10.07	77.85 ± 8.76	4.27 ± 0.71	4.88 ± 1.20
Conval-escent	*BayesAct*		16.63 ± 7.03	14.32 ± 6.61	6.74 ± 0.32	6.78 ± 0.21
	prompt	mind	48.42 ± 12.81	44.32 ± 10.68	5.66 ± 0.69	5.79 ± 0.78
	confer with	mind	18.89 ± 5.79	17.21 ± 6.23	6.68 ± 0.32	6.46 ± 0.42
	command	mind	62.24 ± 7.88	62.44 ± 7.67	5.09 ± 0.58	5.81 ± 0.61
Boss	*BayesAct*		66.60 ± 9.04	68.95 ± 8.83	5.17 ± 0.73	5.01 ± 1.23
	prompt	mind	86.67 ± 8.07	93.08 ± 6.38	3.42 ± 0.83	2.53 ± 1.20
	confer with	mind	62.38 ± 12.50	64.66 ± 16.59	5.47 ± 0.74	4.97 ± 0.44
	command	mind	90.54 ± 6.54	93.43 ± 8.46	3.18 ± 0.98	2.78 ± 1.03

plan-graph), unless the deflection is high, in which case the PwD is more likely to become confused (lose awareness). If she does not advance, she loses awareness. On the other hand, if the PwD is not aware, and is prompted when deflection is low, then she will also move forward (according to the prompt) and gain awareness. However, a high-deflection prompt will again lead to loss of awareness, and to slower progress.

Table 7 shows an example simulation between the agent with the affective identity of "assistant" ($EPA = [1.5, 0.51, 0.45]$) and a client (PwD) with the affective identity of "elder" ($EPA = [1.67, 0.01, -1.03]$). The *BayesAct* agent must learn this identity (shown as $\mathbf{f}_c$ in Table 7) during the interaction if it wants to minimize deflection. We see in this case that the client starts with AW="yes" (1) and does the first two steps, but then stops and is prompted by the agent to rinse his hands. This is the only prompt necessary, the deflection stays low, the agent gets a reasonable estimate of the client identity ($EPA = [2.8, -0.13, -1.36]$, a distance of 1.0). We show example utterances in the table that are "made up" based on our extensive experience working with PwD interacting with a handwashing assistant. Table 8 shows the same client ("elder") but this time the agent always uses the same affective actions: if prompting, it "commands" the user ($EPA = [-0.09, 1.29, 1.59]$) and when not prompting it "minds" the user ($EPA = [0.86, 0.17, -0.16]$). Here we see that the agent prompts cause significant deflection, and this causes the PwD to lose awareness (to become confused) and not make any progress. The handwashing takes much longer, and the resulting interaction is likely much less satisfying.

We modify this *COACH* POMDP model by adding 3D continuous state, action and observation spaces to represent affective identities and behaviours (the normative action bias is *BayesAct*). The social coordination bias is that the PwDs progress through the task is helped by prompting, but only if the deflection is sufficiently low. We investigate the system in simulation using an agent identity of "assistant" ($EPA = [1.5, 0.51, 0.45]$). This "assistant" agent interacts with the following fixed (but unknown to the agent) client identities: "elder" ([1.67, 0.01, −1.03]), "patient" ([0.90, −0.69, −1.05]),

Table 7: Example simulation between the agent and a client (PwD) who holds the affective identity of "elder". Affective actions are chosen by *BayesAct*. Possible utterances for ***agent*** and ***client*** are shown that may correspond to the affective signatures computed.

TURN	CLIENT STATE		ACTION		AGENT EXPECTATION			CLIENT
	AW	*PS*	Prop.	Affect	$\mathbf{f}_c$	*PS*	*AW*	**DEFL.**
initial	1	0	-	-	[0.9,-0.69,-1.05]	0	0.72	-
client	1	0	put on soap ***"[looks at sink]"***	[1.6,0.77,-1.4]	[2.3,-0.77,-1.23]	0.96	0.94	0.23
agent	1	1	- ***"[looks at client]"***	[1.3,0.26,-0.40]	[2.41,-0.81,-1.23]	1.0	≈1.0	1.07
client	1	1	turn on tap ***"oh yes, this is good"***	[2.2,0.90,-1.1]	[2.7,-0.36,-1.37]	3.0	0.99	0.99
agent	1	3	- ***"I'm here to help, Frank"***	[1.3,0.4,0.35]	[2.7,-0.37,-1.38]	3.0	≈1.0	1.47
client	1	3	- ***"this is nice"***	[2.1,0.72,-1.4]	[2.6,-0.34,-1.38]	3.0	0.01	1.14
agent	0	3	rinse hands ***"Great! Let's rinse hands"***	[1.5,0.67,0.06]	[2.6,-0.34,-1.39]	3.0	≈0.0	1.50
client	0	3	rinse hands ***"oh yes, this is good"***	[1.9,0.78,-1.4]	[2.7,-0.31,-1.44]	4.0	0.99	1.11
agent	1	4	- ***"good job Frank"***	[1.6,0.47,-0.13]	[2.7,-0.30,-1.4]	4.0	≈1.0	1.61
client	1	4	turn tap off ***"[looks at tap]"***	[2.0,0.94,-1.3]	[2.6,-0.17,-1.24]	5.9	0.96	1.19
agent	1	6	- ***"This is nice, Frank"***	[1.5,0.56,-0.35]	[2.6,-0.17,-1.2]	6.0	≈1.0	1.56
client	1	6	- ***"Oh yes, good good"***	[2.1,0.86,-1.42]	[2.8,-0.14,-.14]	6.0	≈0.0	1.22
agent	1	6	dry hands ***"[looks at]"***	[1.4,0.66,-0.06]	[2.8,-0.13,-1.36]	6.0	≈0.0	1.55
client	1	6	dry hands ***"all done!"***	[1.94,1.1,-1.9]	-	-	-	1.55
client	1	7	-	-	-	-	-	-

"convalescent" ($[0.3, 0.09, -0.03]$), and "boss"[7] ($[0.48, 2.16, 0.94]$). We compare two policies, in which the affective actions (i.e. how to deliver a prompt) are either computed with *BayesAct* and POMCP-C, or are fixed (as in the current *COACH* system). In both cases, POMCP-C is used to compute a policy for propositional actions (i.e. what prompt to give). We run 10 sets of 10 simulated trials. The results are shown in Table 6. As expected, the fixed policy of "command" ($[-0.09, 1.29, 1.59]$) gives the worst performance in all cases. These results suggest that a fixed affective policy may work for some affective identities, but not for others, whereas the POMCP-C policy can learn and adapt to different client identities.

The difference in Table 6 between *BayesAct* used with POMCP-C and without is not significant. In 10 out of 16 tests, POMCP-C allows for action choices that lead to fewer interactions, and does better on average. Perhaps more interestingly, this estimate of error can be used to evaluate the quality of the action bias within the given interaction. If the POMCP-C model starts doing worse on average than a fixed policy, it is an indication that the action bias is not very good, and that there is a possible misalignment between the agent and the patient.

4.4 8D Intersection Problem

POMCP-C can be generalized to non-affective domains where *action biases* naturally arise. In the 8D Intersection Problem [9], a robot agent's task is to navigate a 2D space to reach a goal, by avoiding a

[7] Many persons with Alzheimer's disease think of themselves in terms of some past identity or role.

Table 8: Example simulation between the agent and a client (PwD) who holds the affective identity of "elder". Affective actions were fixed: if prompting, it "commands" the user and when not prompting it "minds" the user.

TURN	CLIENT STATE		ACTION		AGENT EXPECTATION			CLIENT
	AW	*PS*	Prop.	Affect	$\mathbf{f}_c$	*PS*	*AW*	**DEFL.**
initial	1	0	-	-	[0.9,-69,-1.05]	0	0.72	-
client	1	0	put on soap ***"[looks at sink]"***	[1.6,0.77,-1.4]	[2.3,-0.77,-1.23]	0.96	0.94	0.23
agent	1	1	- ***"[looks at client]"***	[0.85,0.17,-0.16]	[2.41,-0.81,-1.23]	1.0	≈1.0	1.34
client	1	1	turn on tap ***"oh yes, this is good"***	[2.3,0.90,-1.19]	[2.62,-0.42,-1.43]	2.98	0.99	1.21
agent	1	3	- ***"[looks at client]"***	[0.85,0.17,-0.16]	[2.7,-0.42,-1.5]	3.0	≈1.0	1.86
client	1	3	- ***"oh yes, this is good"***	[2.2,0.79,-1.47]	[2.6,-0.30,-1.4]	3.0	≈0.0	1.56
agent	0	3	rinse hands ***"Rinse your hands!"***	[-0.1,1.29,1.59]	[2.6,-0.30,-1.4]	3.0	≈0.0	4.11
client	0	3	- ***"[looks at sink]"***	[1.9,1.4,-1.7]	[2.5,-0.30,-1.3]	3.0	≈0.0	2.90
agent	0	3	rinse hands ***"Rinse your hands!"***	[-0.1,1.29,1.59]	[2.5,-0.29,-1.3]	3.0	≈0.0	5.80
client	0	3	- ***"[looks at sink]"***	[1.9,0.97,-1.9]	[2.4,-0.27,-1.26]	3.0	0.02	4.28
agent	0	3	rinse hands	[-0.1,1.29,1.59]	[2.4,-0.26,-1.27]	3.0	0.02	7.05

...continues for 85 more steps until client finally finishes ...

moving obstacle. The state space consists of 8-tuples $(a_x, a_y, a_{vx}, a_{vy}, ob_x, ob_y, ob_{vx}, ob_{vy}) \in \mathbb{R}^8$, where the first four values give the agent's position and velocity in 2D, and the last four values give the obstacle's position and velocity in 2D. The agent and obstacle are circular disks of radius 1. The obstacles starts at $ob_x = 6.0\ m$ and a random $ob_y \in [-8.0, 8.0]$, and moves vertically[8] with fixed velocity $(ob_{vx}, ob_{vy}) = (0.0\ m/s, 1.0\ m/s)$. The agent starts with $(a_x, a_y, a_{vx}, a_{vy}) = (0.0\ m, 0.0\ m, 1.0\ m/s, 0.0\ m/s)$, and at each time step, it decides whether to accelerate by $\pm 1.0\ m/s^2$ on the horizontal axis, to reach the goal located at $(9.0\ m, 0.0\ m)$. The agent has no knowledge of the position of the obstacle unless they are less than 4 m apart. The discount factor is 0.95, and the reward is $+10$ for reaching the goal and -10 for colliding with the obstacle. Thus, the problem has an 8D continuous state space, a discrete (binary) action space, and an 8D continuous observation space (a noisy measurement of the state).

We ran 1000 simulated trials with the following settings. The belief state of the agent consists of 50,000 particles, and is initialised to an 8D Gaussian with mean $(0.0, 0.0, 1.0, 0.0, 6.0, 0.0, 0.0, 1.0)$ and diagonal covariance $(10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}, 8.0, 10^{-4}, 10^{-4})$. When the agent and obstacle are more than 4 m apart, the agent receives a uniformly random observation of the obstacle's position and velocity. Otherwise, the mean of the observation model $P(o|x)$ is always the state x with variance 10^{-4}. In addition, the model incorporates small white noise. With the POMCP-C parameters $\delta_o = 0.5$ and $Timeout = 15s$, our algorithm achieves an average discounted reward of 5.9, compared to 5.0 by Brechtel *et al*'s point-based backups.

To further demonstrate how POMCP-C handles continuous action spaces, we devise a continuous-action version of this problem, where the agent can sample actions from the action bias $\mathcal{N}(1.0, \sigma) \cup \mathcal{N}(-1.0, \sigma)$, σ being a settable parameter. The number of particles, initial belief state and observation model remain the same. With $\sigma = 0.4$, $\delta_a = 0.25$, $\delta_o = 0.5$, $N_A^{max} = 15$ and $Timeout = 400$, POMCP-C achieves an average discounted reward of 7.8. This is a dramatic improvement compared to

[8] Whenever $ob_y > 8.0\ m$, it is reset to -8.0.

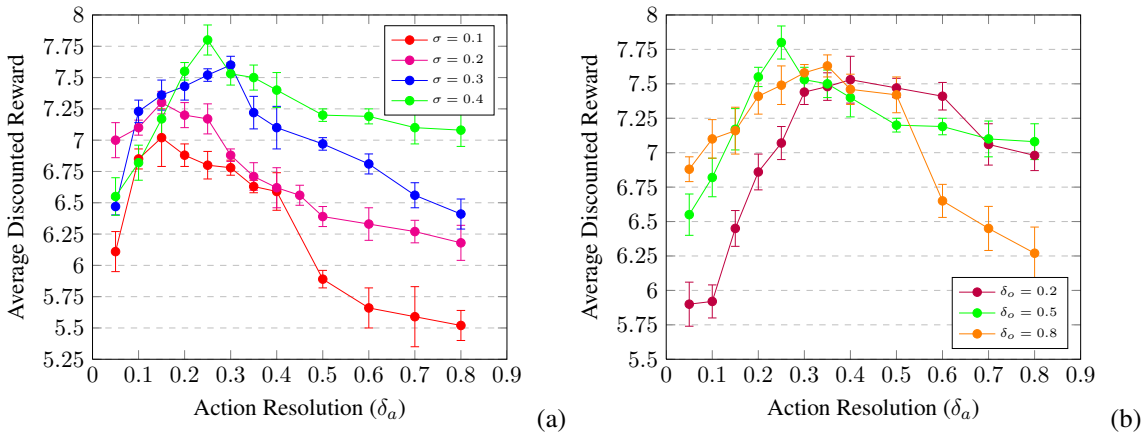

Figure 4: 8D Intersection Problem (continuous actions). $\sigma = 0.4$, $\delta_o = 0.5$, $N_A^{max} = 15$, $Timeout = 400$ unless otherwise noted.

the discrete-action version, as the agent can fine-tune its acceleration to avoid the moving obstacle more often. Figure 4(a) and (b) show how the average discounted reward varies with δ_a, for different fixed values of σ and δ_o, respectively. We see that the average reward increases rapidly with increasing δ_a (as the number of actions in the tree decreases, leading to more search per action), but peaks and declines slowly because important actions are not distinguished anymore.

5 Related Work

Damasio has convincingly argued, both from a functional and neurological standpoint, for emotions playing a key role in decision making and for human social action [13]. His *Somatic Marker Hypothesis* is contrasted against the Platonic "high-reason" view of intelligence, in which pure rationality is used to make decisions. Damasio argues that, because of the limited capacity of working memory and attention, the Platonic view will not work. Instead, learned neural markers focus attention on actions that are likely to succeed, and act as a neural bias allowing humans to work with fewer alternatives. These *somatic markers* are "cultural prescriptions" for behaviours that are "rational relative to the social conventions and ethics" ([13], p200).

LeDoux [39] argues the same thing from an evolutionary standpoint. He theorises that the subjective feeling of emotion must take place at both unconscious and conscious levels in the brain, and that consciousness is the ability to relate stimuli to a sense of identity, among other things. This is not a new idea. Writing over a century ago, William James pointed to self-reference as the key to consciousness [34]. That is, *thinking* and *feeling* can only exist consciously in relation to an individual, a *self* [37].

With remarkably similar conclusions coming from a more functional (economic) viewpoint, Kahneman has demonstrated that human emotional reasoning often overshadows, but is important as a guide for, cognitive deliberation [36]. Kahneman presents a two-level model of intelligence, with a fast/normative/reactive/affective mechanism being the "first on the scene", followed by a slow/cognitive/deliberative mechanism that operates if sufficient resources are available. Akerlof and Kranton attempt to formalise *fast thinking* by incorporating a general notion of identity into an economic model (utility function) [2]. Earlier work on *social identity theory* foreshadowed this economic model by noting that simply assigning group membership increases individual cooperation [33, 61]. Other authors have also confirmed that group membership influences individual choice (e.g. [11]). This work has been contested by the counter-argument that it is not the group membership that increases cooperation, but rather that the

group membership increases individual's beliefs that others will cooperate (see [38]). *BayesAct* unifies these two views, since beliefs about cooperation are tightly tied to beliefs about identities in the group. If a person is told they are in a group of robbers, they will have the belief that others will defect because that is the prescription for the identity of robber, even when interacting with other robbers.

The idea that unites Kahneman, LeDoux, and Damasio (and others) is the tight connection between emotion and action. These authors, from very different fields, propose emotional reasoning as a "quick and dirty", yet absolutely necessary, guide for cognitive deliberation. The neurological underpinnings of this connection are discussed by Zhu and Thagard [64] who, following LeDoux [39], point to the amygdala as being the "hub" the wheel of emotional processing in the brain, and discuss how emotion plays an important role in both the generation, and in the execution of action. They discuss two neural pathways from the sensory thalamus to amygdala that are used in action generation: the direct "low road", and the more circuitous "high road" that makes a stop in the sensory cortex. While the low road enables fast, pre-programmed, reactive responses, the high-road enables a more careful evaluation of the situation. The two pathways complement each other, with the default option opting for more potentially life-saving false alarms than the high road. *ACT* gives a functional account of the quick pathway as sentiment encoding prescriptive behaviour, while *BayesAct* shows how this account can be extended with a slow pathway that enables exploration and planning away from the prescription.

Our work fits well into a wide body of work on *affective computing* (AC) [51, 53], with a growing focus on socio-cultural agents (e.g. [15]). In AC, emotions are usually inferred based on cognitive appraisals (e.g. a thwarted goal causes anger), and these are used to guide action through a set of "coping" mechanisms. Emotions are usually framed following the rationalistic view proposed by Simon as "interrupts" to cognitive processing [59]. Gratch and Marsella [23] are possibly the first to propose a concrete computational mechanism for coping. Building on the work of Smith and Lazarus [60], they propose a five stage process wherein beliefs, desires, plans and intentions are first formulated, and upon which emotional appraisals are computed. Coping strategies then use a set of *ad hoc* rules by modifying elements of the model such as probabilities and utilities, or by modifying plans or intentions. Si *et al.* [57] compute emotional appraisals from utility measures (including beliefs about other agent's utilities, as in an I-POMDP [22]), they leave to future work *"how emotion affects the agents decision-making and belief update processes"* ([57] section 8). Goal prioritization using emotional appraisals have been investigated [4, 19, 32, 42, 47, 47], as have normative multi-agent systems (NorMAS) [7]. There has been recent work on facial expressions in PD games, showing that they can significantly affect the outcomes [12, 14].

Most approaches to emotional action guidance only give broad action guides in extreme situations, leaving all else to the cognitive faculties. *BayesAct* specifies one simple coping mechanism: minimizing inconsistency in continuous-valued sentiment. This, when combined with mappings describing how sentiments are appraised from events and actions, can be used to prescribe actions that maximally reduce inconsistency. These prescriptions are then used as guides for higher-level cognitive (including rational) processing and deliberation. *BayesAct* therefore provides an important step in the direction of building models that integrate "cognitive" and "affective" reasoning.

The affective component of *BayesAct* (e.g. $\mathbf{f}, \boldsymbol{\tau}, \mathbf{b_a}$) describes fast, heuristic, everyday human interaction: it is what humans use to "get along" in a social world. The propositional component, on the other hand (e.g. $\mathbf{x}, a$), describes more traditional cognitive artificial intelligence. *BayesAct* agents are free to use $\mathbf{X}$ to model other agents at any level of detail (including as full POMDPs [17]). Such more complex modeling will allow agents to reason about how other agents are reasoning (cognitively) about them, etc. Nevertheless, even such crafty and devious agents will need to follow the norms of the society they are trying to manipulate, otherwise other agents will be sure to suspect something is up!

BayesAct requires anytime techniques for solving large continuous POMDPs with non-Gaussian beliefs. While [63] describe exact symbolic methods for handling such problems, it is a challenge to scale their methods to the size of problems we are considering in this paper. There has been much recent effort

in solving continuous POMDPs with Gaussian beliefs (e.g. [16, 20, 49]), but these are usually in robotics motion planning where such approximations are reasonable. Point-based methods [56] have yielded solutions for continuous POMDPs for small domains [6, 9, 52], but they are not anytime and scalability is an issue, although recent work in parallel versions of point-based algorithms may lead to greater scalability [40]. Point-based methods (e.g. [52]) require the value function to be closed under the Bellman operator, which is not possible for *BayesAct*. Continuous Perseus, for example [52], is an approximate point-based algorithm for computing value functions (sets of alpha functions) for domains with continuous states. However, the value function itself must be closed under the Bellman backup operator to make the computation tractable, requiring a linear-Gaussian transition function (which we do not have). Even if a linearisation of the ACT equations was found, the explosion of the number of Gaussian mixtures requires contraction operators that further complicate the approximation. Other representations of alpha functions are as policy graphs [6], which does not work for continuous observations, or as decision trees [9], which we compared against in Section 4.4.

Approximate techniques using Monte-Carlo tree search (MCTS) methods have seen more scalability success [10], and are anytime. POMCP [58] uses MCTS to efficiently solve POMDPs with continuous state spaces. By design, POMCP is unable to handle models with continuous action spaces, such as *BayesAct*. POMCoP uses POMCP to guide a sidekick's actions during a cooperative video game [44]. While this game has many similarities to CoRobots, it does not have continuous actions and restricts agent types to a small and countable set. POMCoP also uses an action bias, in this case it predicts the human's movements in the video game according to their type (this would be equivalent to our social coordination bias). Recent proposals for large-scale POMDPs have included methods to leverage structure in multi-agent teams [3]. Guez *et al.* [24] present a variant of POMCP called BAMCP for solving Bayes-adaptive Markov decision processes (BAMDPs), which have continuous state, but deterministic (constant) dynamics. BAMDPs are POMDPs in which the continuous state (the model parameters) remains constant. BAMCP works by selecting a single sample from the distribution over models at the start of a simulation, so it is not general enough to work for our POMDPs, which have continuous states but not constant dynamics.

MCTS methods are more appealing for *BayesAct* than other solvers because: (1) MCTS does not require a computation of the value function over the continuous state space and non-linear dynamics; (2) MCTS provides an anytime "quick and dirty" solution that corresponds naturally to our interpretation of the "fast thinking" heuristic.

6 Conclusion

We have studied an anytime planning method for a class of POMDP models of affective interactions, *BayesAct*, in which culturally shared sentiments are used to provide normative action guidance. *BayesAct* is an exciting new development in artificial intelligence that combines affective computing, sociological theory, and probabilistic modeling. The large state, action and observation space of these POMDPs, along with the non-linear dynamics, leads us to a Monte-Carlo Tree Search (MCTS) method based on the POMCP algorithm. We show results of this method on two simulated social dilemmas, demonstrating manipulative behaviours. We also present an assistive device for persons with dementia that can plan strategies of affective prompts or cues. Finally, we show how our MCTS algorithm can tackle other domains with impressive results.

References

[1] K. J. Åström. Optimal control of Markov decision processes with incomplete state estimation. *J. Math. Anal. App.*, 10:174–205, 1965.

[2] George A. Akerlof and Rachel E. Kranton. Economics and identity. *The Quarterly Journal of Economics*, CXV(3), August 2000.

[3] Christopher Amato and Frans A. Oliehoek. Scalable planning and learning for multiagent pomdps. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, TX, January 2015.

[4] Dimitrios Antos and Avi Pfeffer. Using emotions to enhance decision-making. In *Proc. International Joint Conferences on Artificial Intelligence*, Barcelona, Spain, 2011.

[5] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[6] Haoyu Bai, David Hsu, Wee Sun Lee, and Vien A. Ngo. Monte-Carlo value iteration for continuous-state POMDPs. In *Workshop on the Algorithmic Foundations of Robotics*, pages 175–191, 2010.

[7] Tina Balke, Célia da Costa Pereira, Frank Dignum, Emiliano Lorini, Antonino Rotolo, Wamberto Vasconcelos, and Serena Villata. Norms in MAS: Definitions and Related Concepts. In Giulia Andrighetto, Guido Governatori, Pablo Noriega, and Leendert W. N. van der Torre, editors, *Normative Multi-Agent Systems*, volume 4 of *Dagstuhl Follow-Ups*, pages 1–31. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2013.

[8] Craig Boutilier, Thomas Dean, and Steve Hanks. Decision theoretic planning: Structural assumptions and computational leverage. *Journal of Artifical Intelligence Research*, 11:1–94, 1999.

[9] Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. Solving continuous pomdps: Value iteration with incremental learning of an efficient space representation. In *Proc. of Intl. Conference on Machine Learning (ICML)*, 2013.

[10] C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of Monte Carlo tree search methods. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4(1):1–43, March 2012.

[11] Gary Charness, Luca Rigotti, and Aldo Rustichini. Individual behavior and group membership. *The American Economic Review*, 97(4):pp. 1340–1352, 2007.

[12] Rosaria Conte and Cristiano Castelfranchi. *Cognitive and Social Action*. UCL Press, London, 1995.

[13] Antonio R. Damasio. *Descartes' error: Emotion, reason, and the human brain*. Putnam's sons, 1994.

[14] Celso M. de Melo, Peter Carnevale, Stephen Read, Dimitrios Antos, and Jonathan Gratch. Bayesian model of the social effects of emotion in decision-making in multiagent systems. In *Proc. International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Valencia, Spain, 2012.

[15] Nick Degens, Gert Jan Hofstede, John McBreen, Adrie Beulens, Samuel Mascarenhas, Nuno Ferreira, Ana Paiva, and Frank Dignum. Creating a world for socio-cultural agents. In Tibor Bosse, Joost Broekens, Joao Dias, and Janneke van der Zwaan, editors, *Emotion Modeling: Towards Pragmatic Computational Models of Affective Processes*, number 8750 in Lecture Notes in Artificial Intelligence. Springer, 2014.

[16] Marc Peter Deisenroth and Jan Peters. Solving nonlinear continuous state-action-observation pomdps for mechanical systems with gaussian noise. In *Proceedings of the European Workshop on Reinforcement Learning (EWRL)*, 2012.

[17] Prashant Doshi and Piotr Gmytrasiewicz. Monte-Carlo sampling methods for approximating interactive POMDPs. *Journal of Artificial Intelligence Research*, 34:297–337, 2009.

[18] Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors. *Sequential Monte Carlo in Practice*. Springer-Verlag, 2001.

[19] Magy Seif El-Nasr, John Yen, and Thomas R. Ioerger. Flame - fuzzy logic adaptive model of emotions. *Autonomous Agents and Multiagent Systems*, pages 219–257, 2000.

[20] Tom Erez and William D. Smart. A scalable method for solving high-dimensional continuous pomdps using local approximation. In *Proc. of the 26th Conf. in Uncertainty in Artificial Intelligence (UAI)*, 2010.

[21] Jeremiah T. Folsom-Kovarik, Gita Sukthankar, and Sae Schatz. Tractable POMDP representations for intelligent tutoring systems. *ACM Trans. Intell. Syst. Technol.*, 4(2):29:1–29:22, April 2013.

[22] Piotr Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

[23] Jonathan Gratch and Stacy Marsella. A domain-independent framework for modeling emotion. *Cognitive Systems Research*, 5(4):269 – 306, 2004.

[24] Arthur Guez, David Silver, and Peter Dayan. Scalable and efficient Bayes-adaptive reinforcement learning based on Monte-Carlo tree search. *Journal of Artificial Intelligence Research*, 48, 2013.

[25] David R. Heise. *Expressive Order: Confirming Sentiments in Social Actions*. Springer, 2007.

[26] David R. Heise. *Surveying Cultures: Discovering Shared Conceptions and Sentiments*. Wiley, 2010.

[27] David R. Heise. Modeling interactions in small groups. *Social Psychology Quarterly*, 76:52–72, 2013.

[28] Jesse Hoey, Craig Boutilier, Pascal Poupart, Patrick Olivier, Andrew Monk, and Alex Mihailidis. People, sensors, decisions: Customizable and adaptive technologies for assistance in healthcare. *ACM Trans. Interact. Intell. Syst.*, 2(4):20:1–20:36, January 2012.

[29] Jesse Hoey and Tobias Schröder. Bayesian affect control theory of self. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015.

[30] Jesse Hoey, Tobias Schröder, and Areej Alhothali. Affect control processes: Intelligent affective interaction using a partially observable Markov decision process. http://arxiv.org/abs/1306.5279, 2013.

[31] Jesse Hoey, Tobias Schröder, and Areej Alhothali. Bayesian affect control theory. In *Proc. of the 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction (ACII 2013)*, 2013.

[32] Eric Hogewoning, Joost Broekens, Jeroen Eggermont, , and Ernst G.P. Bovenkamp. Strategies for affect-controlled action-selection in soar-rl. In J. Mira and J.R. Àlvarez, editors, *IWINAC*, volume 4528 Part II of *LNCS*, pages 501–510, 2007.

[33] Michael A. Hogg. Social identity theory. In Peter J. Burke, editor, *Contemporary Social Psychological Theories*, chapter 6, pages 111–136. Stanford University Press, 2006.

[34] William James. *Principles of Psychology*. Holt, New York, 1890.

[35] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[36] Daniel Kahneman. *Thinking, Fast and Slow*. Doubleday, 2011.

[37] John F. Kihlstrom. The cognitive unconscious. *Science*, 237:1445–1452, September 1987.

[38] Peter Kollock. Social dilemmas: the anatomy of cooperation. *Annual Review of Sociology*, 24:183–214, 1998.

[39] Joseph LeDoux. *The emotional brain: the mysterious underpinnings of emotional life*. Simon and Schuster, New York, 1996.

[40] Taekhee Lee and Young J. Kim. GPU-based motion planning under uncertainties using POMDP. In *Proc. Intl Conf. on Robotics and Automation (ICRA)*, Karlsruhe, DE, 2013.

[41] Luyuan Lin, Stephen Czarnuch, Aarti Malhotra, Lifei Yu, Tobias Schröder, and Jesse Hoey. Affectively aligned cognitive assistance using bayesian affect control theory. In *Proc. of International Workconference on Ambient Assisted Living (IWAAL)*, pages 279–287, Belfast, UK, December 2014. Springer.

[42] Christine Laetitia Lisetti and Piotr Gmytrasiewicz. Can a rational agent afford to be affectless? a formal approach. *Applied Artificial Intelligence*, 16(7-8):577–609, 2002.

[43] W. S. Lovejoy. A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, 28:47–66, 1991.

[44] Owen Macindoe, Leslie Pack Kaelbling, , and Tomás Lozano-Pérez. Pomcop: Belief space planning for sidekicks in cooperative games. In *AIIDE 2012*, 2012.

[45] Neil. J. MacKinnnon and Dawn T. Robinson. 25 years of research in affect control theory. *Advances in Group Processing*, 31, 2014.

[46] Aarti Malhotra, Celia Yu, Tobias Schröder, and Jesse Hoey. An exploratory study into the use of an emotionally aware cognitive assistant. In *Proc. AAAI Workshop on artificial intelligence applied to assistive technologies and smart environments (ATSE)*, 2015.

[47] Robert P. Marinier III and John E. Laird. Emotion-driven reinforcement learning. In *Proc. of 30th Annual Meeting of the Cognitive Science Society*, pages 115–120, Washington, D.C., 2008.

[48] Charles E. Osgood, William H. May, and Murray S. Miron. *Cross-Cultural Universals of Affective Meaning*. University of Illinois Press, 1975.

[49] Sachin Patil, Gregory Kahn, Michael Laskey, John Schulman, Ken Goldberg, and Pieter Abbeel. Scaling up gaussian belief space planning through covariance-free trajectory optimization and automatic differentiation. In *Proc. Workshop on Algorithmic Foundations of Robotics - WAFR*, 2014.

[50] Fernando Perez-Cruz. Kullback-Leibler divergence estimation of continuous distributions. In *IEEE International Symposium on Information Theory (ISIT)*, pages 1666–1670, July 2008.

[51] Rosalind W. Picard. *Affective Computing*. MIT Press, Cambridge, MA, 1997.

[52] Josep M. Porta, Nikos Vlassis, Matthijs T.J. Spaan, and Pascal Poupart. Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research*, 7:2329–2367, 2006.

[53] Klaus R. Scherer, Tanja Banziger, and Etienne Roesch. *A Blueprint for Affective Computing*. Oxford University Press, 2010.

[54] Tobias Schröder, Janine Netzel, Carsten Schermuly, and Wolfgang Scholl. Culture-constrained affective consistency of interpersonal behavior: A test of affect control theory with nonverbal expressions. *Social Psychology*, 44:47–58, 2013.

[55] Tobias Schröder and Wolfgang Scholl. Affective dynamics of leadership: An experimental test of affect control theory. *Social Psychology Quarterly*, 72:180–197, 2009.

[56] Guy Shani, Joelle Pineau, and R. Kaplow. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 2012.

[57] Mei Si, StacyC. Marsella, and DavidV. Pynadath. Modeling appraisal in theory of mind reasoning. *Autonomous Agents and Multi-Agent Systems*, 20(1):14–31, 2010.

[58] David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. In *Proc. Neural Information Processing Systems (NIPS)*, December 2010.

[59] Herbert A. Simon. Motivational and emotional controls of cognition. *Psychological Review*, 74:29–39, 1967.

[60] C. Smith and R. Lazarus. Emotion and adaptation. In Pervin, editor, *Handbook of personality: Theory & research*, pages 609–637. Guilford Press, New York, 1990.

[61] Henri Tajfel and John C. Turner. An integrative theory of intergroup conflict. In Stephen Worchel and William Austin, editors, *The social psychology of intergroup relations*. Brooks/Cole, Monterey, CA, 1979.

[62] Jason D. Williams and Steve Young. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):393–422, 2006.

[63] Zahra Zamani, Scott Sanner, and C. Fang. Symbolic dynamic programming for continuous state and action mdps. In *Proc. 26th AAAI Conf. Artificial Intelligence*, Toronto, Canada, 2012.

[64] Jing Zhu and Paul Thagard. Emotion and action. *Philosophical Psychology*, 15(1), 2002.

Appendix A Prisoner's Dilemma Full Results

γ			timeout						
			1	2	5	10	30	60	120
0.90	mean	agent	10.29 ± 0.38	10.33 ± 0.42	10.55 ± 0.28	10.44 ± 0.34	10.72 ± 0.18	10.76 ± 0.20	10.96 ± 0.18
		client	7.10 ± 3.75	6.70 ± 4.21	4.50 ± 2.80	5.65 ± 3.44	2.80 ± 1.82	2.45 ± 1.96	0.45 ± 1.79
	mean	agent	10.58 ± 0.10	10.66 ± 0.08	10.78 ± 0.22	10.69 ± 0.19	10.85 ± 0.21	10.87 ± 0.24	11.00 ± 0.00
	(last 10)	client	4.20 ± 1.03	3.40 ± 0.84	2.20 ± 2.20	3.10 ± 1.91	1.50 ± 2.07	1.30 ± 2.36	0.00 ± 0.00
	median	agent	10.00	10.00	11.00	10.00	11.00	11.00	11.00
		client	10.00	10.00	0.00	10.00	0.00	0.00	0.00
	median	agent	11.00	11.00	11.00	11.00	11.00	11.00	11.00
	(last 10)	client	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table A1: PD experiments with client strategy: (co) and discount $\gamma = 0.9$

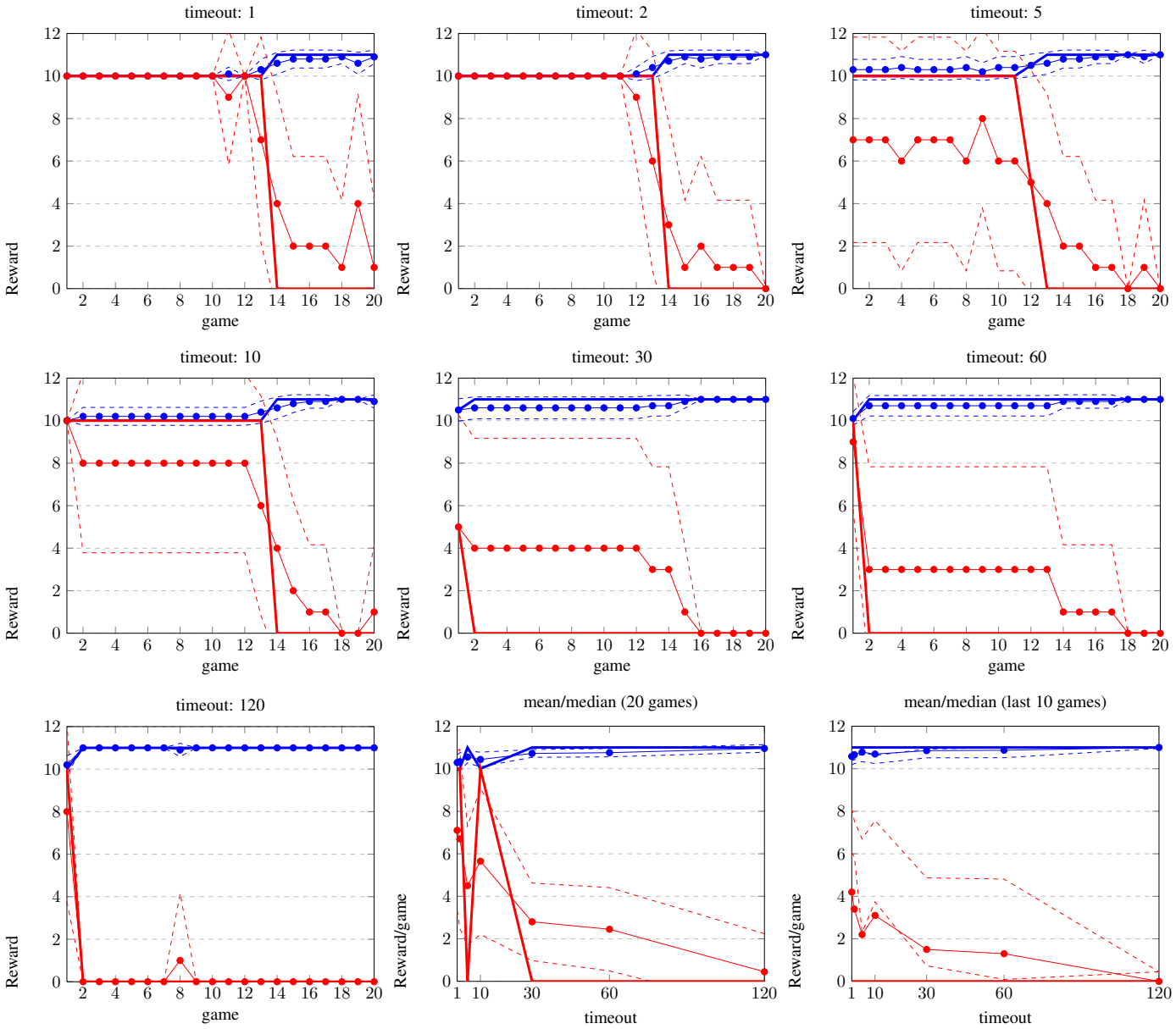

Figure A1: PD experiments with client strategy: (co) and discount $\gamma = 0.9$ Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			timeout						
			1	2	5	10	30	60	120
0.90	mean	agent	0.67 ± 0.21	0.78 ± 0.24	0.85 ± 0.20	0.89 ± 0.23	0.93 ± 0.16	0.94 ± 0.16	0.93 ± 0.21
		client	4.35 ± 2.11	3.20 ± 2.42	2.50 ± 2.04	2.10 ± 2.31	1.70 ± 1.63	1.60 ± 1.57	1.75 ± 2.15
	mean	agent	0.75 ± 0.16	0.86 ± 0.13	0.94 ± 0.10	0.96 ± 0.07	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	(last 10)	client	3.50 ± 1.58	2.40 ± 1.35	1.60 ± 0.97	1.40 ± 0.70	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	median	agent	1.00	1.00	1.00	1.00	1.00	1.00	1.00
		client	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	median	agent	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	(last 10)	client	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table A2: PD experiments with client strategy: (de) and discount $\gamma = 0.9$

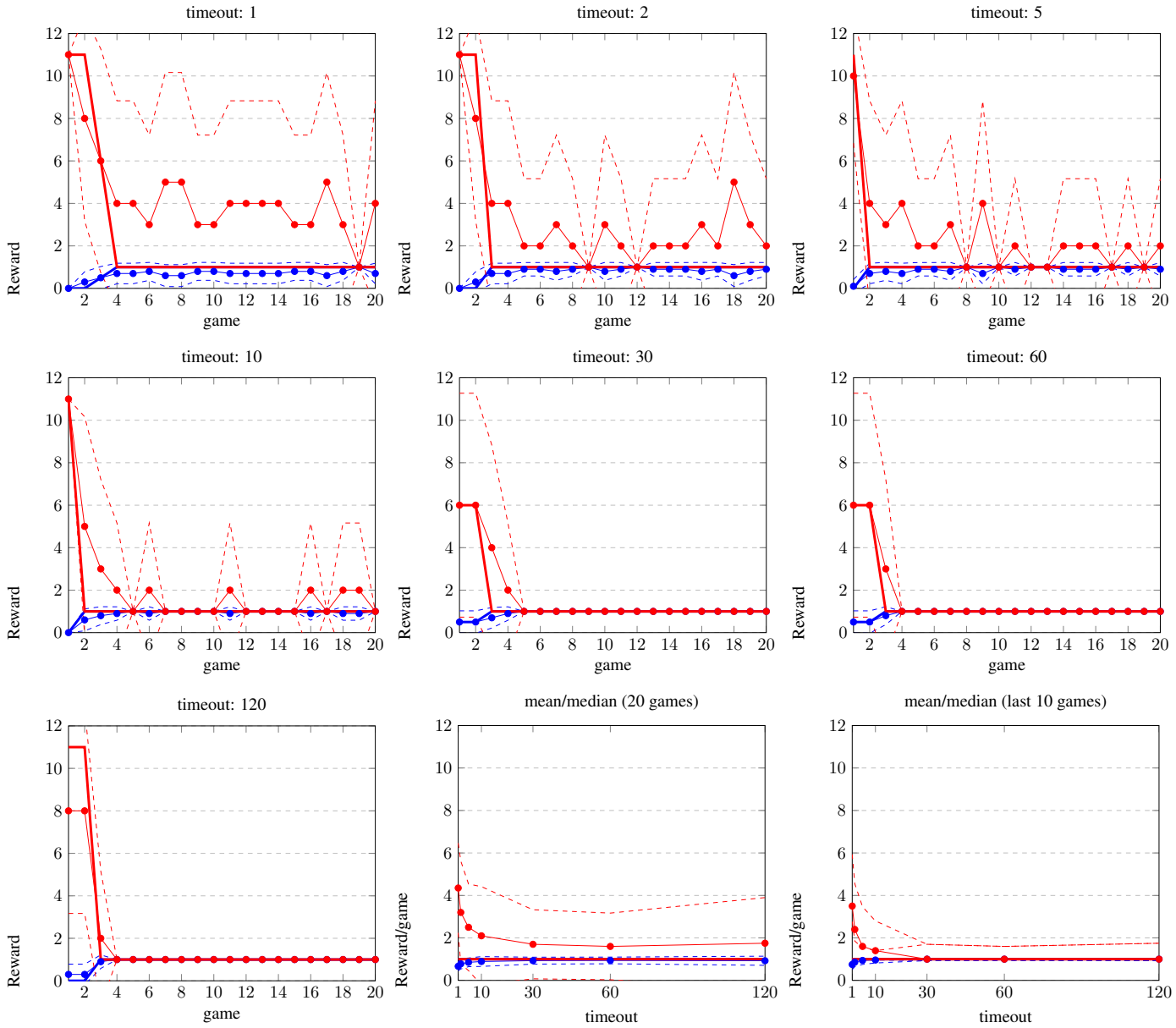

Figure A2: PD experiments with client strategy: (de) and discount $\gamma = 0.9$ Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			timeout						
			1	2	5	10	30	60	120
0.90	mean	agent	1.50 ± 2.92	1.58 ± 2.89	1.81 ± 2.89	1.83 ± 2.89	1.92 ± 2.97	1.92 ± 2.92	1.88 ± 2.89
		client	5.90 ± 2.85	5.05 ± 3.47	2.80 ± 2.26	2.55 ± 2.68	1.70 ± 1.17	1.75 ± 1.65	2.15 ± 2.25
	mean	agent	0.65 ± 0.14	0.79 ± 0.11	0.93 ± 0.07	0.99 ± 0.03	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	(last 10)	client	4.50 ± 1.35	3.10 ± 1.10	1.70 ± 0.67	1.10 ± 0.32	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	median	agent	1.00	1.00	1.00	1.00	1.00	1.00	1.00
		client	5.50	1.00	1.00	1.00	1.00	1.00	1.00
	median	agent	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	(last 10)	client	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table A3: PD experiments with client strategy: (to) and discount $\gamma = 0.9$

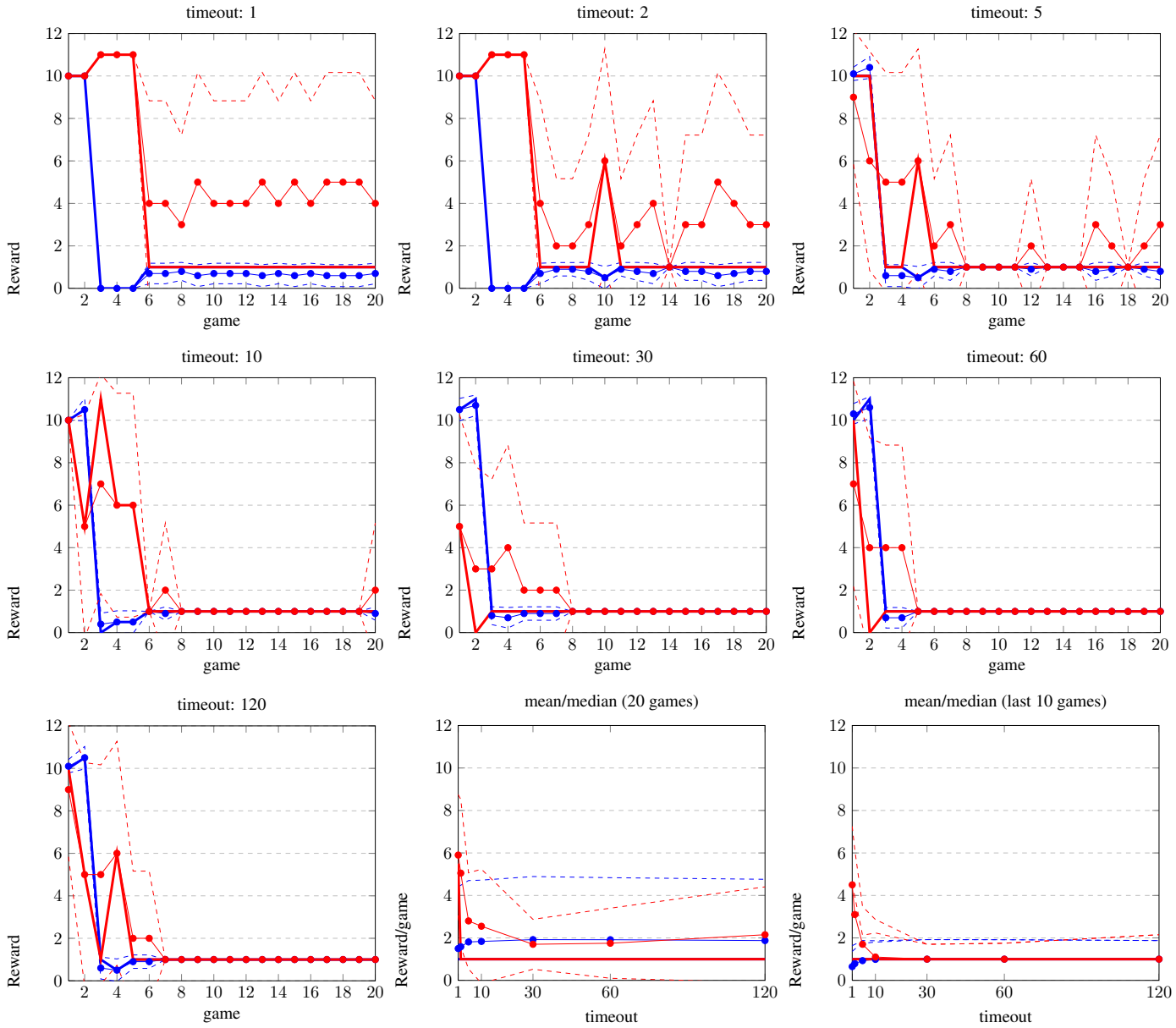

Figure A3: PD experiments with client strategy: (to) and discount $\gamma = 0.9$ Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			timeout						
			1	2	5	10	30	60	120
0.90	mean	agent	7.86 ± 3.23	7.79 ± 3.02	4.74 ± 2.63	3.57 ± 2.51	2.27 ± 2.08	2.67 ± 2.09	1.64 ± 2.24
		client	7.69 ± 3.18	7.35 ± 3.20	4.19 ± 2.39	3.02 ± 1.93	1.72 ± 0.51	2.12 ± 0.88	1.08 ± 0.46
	mean	agent	5.71 ± 1.44	5.58 ± 1.13	2.95 ± 2.01	2.20 ± 1.89	1.55 ± 1.74	1.65 ± 1.39	1.00 ± 0.00
	(last 10)	client	5.38 ± 1.70	4.70 ± 1.11	2.29 ± 1.56	1.87 ± 1.50	1.44 ± 1.39	1.43 ± 0.93	1.00 ± 0.00
	median	agent	10.00	10.00	1.00	1.00	1.00	1.00	1.00
		client	10.00	10.00	1.00	1.00	1.00	1.00	1.00
	median	agent	10.00	5.50	1.00	1.00	1.00	1.00	1.00
	(last 10)	client	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table A4: PD experiments with client strategy: (tt) and discount $\gamma = 0.9$

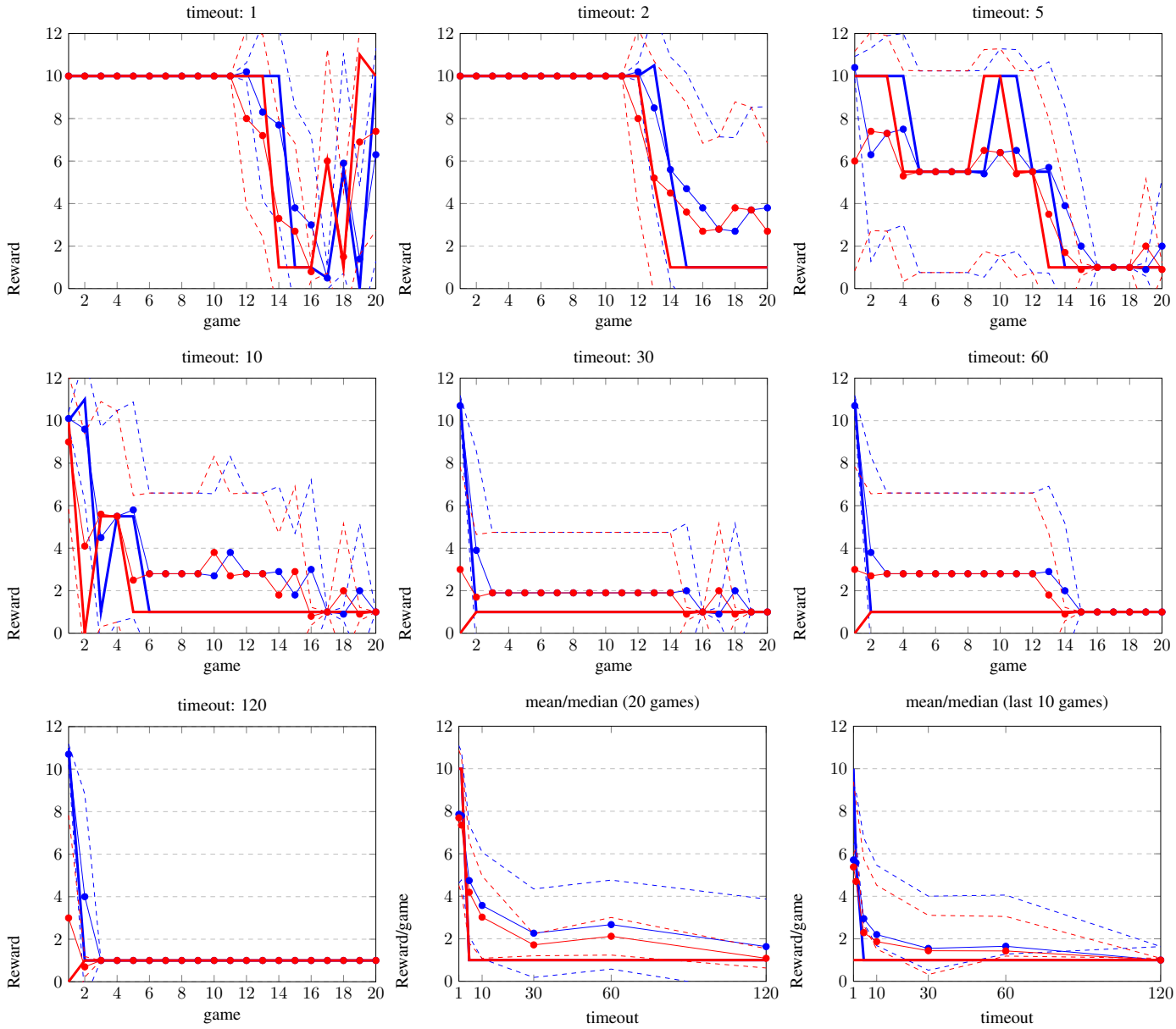

Figure A4: PD experiments with client strategy: (tt) and discount $\gamma = 0.9$ Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			timeout						
			1	2	5	10	30	60	120
0.90	mean	agent	8.70 ± 2.28	8.46 ± 2.54	3.37 ± 2.78	6.18 ± 2.95	3.85 ± 2.61	3.44 ± 2.71	3.98 ± 2.48
		client	7.21 ± 3.54	7.13 ± 3.63	1.55 ± 0.87	4.64 ± 2.83	2.65 ± 1.10	2.29 ± 1.10	2.83 ± 1.24
	mean	agent	7.39 ± 1.17	6.91 ± 1.07	1.96 ± 1.35	3.92 ± 2.63	2.57 ± 3.32	2.03 ± 2.18	2.69 ± 2.86
	(last 10)	client	4.42 ± 1.40	4.27 ± 1.01	1.30 ± 0.58	2.49 ± 1.54	2.13 ± 2.39	1.59 ± 1.26	1.92 ± 1.61
	median	agent	10.00	10.00	1.00	10.00	1.00	1.00	1.00
		client	10.00	10.00	1.00	1.00	1.00	1.00	1.00
	median	agent	10.00	10.00	1.00	1.00	1.00	1.00	1.00
	(last 10)	client	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table A5: PD experiments with client strategy: (t2) and discount $\gamma = 0.9$

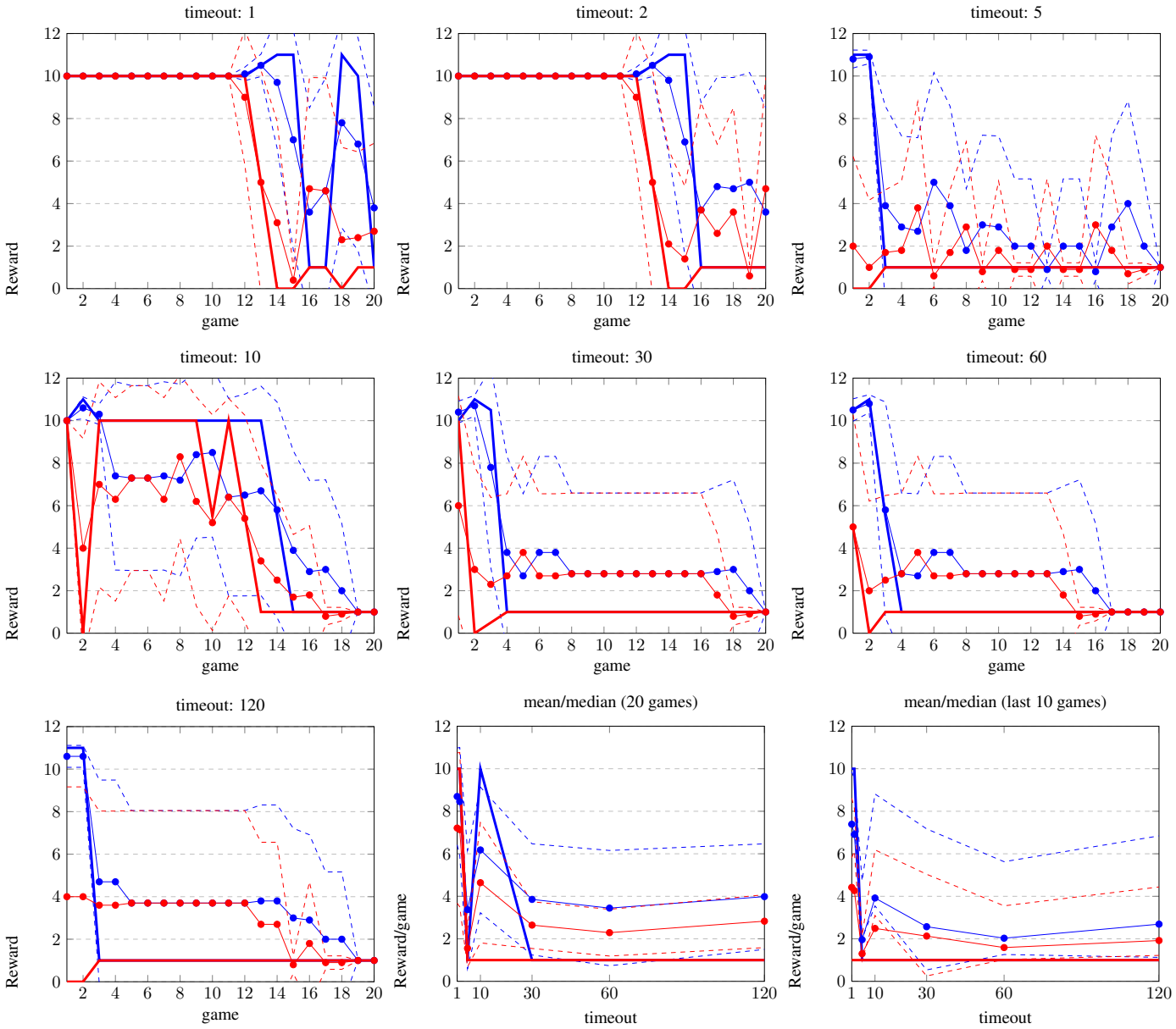

Figure A5: PD experiments with client strategy: (t2) and discount $\gamma = 0.9$ Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			timeout						
			1	2	5	10	30	60	120
0.90	mean	agent	7.26 ± 3.89	7.21 ± 4.13	4.47 ± 2.46	5.75 ± 2.89	2.75 ± 2.15	3.48 ± 2.17	1.72 ± 2.35
		client	7.21 ± 3.50	7.21 ± 3.64	4.42 ± 2.15	5.42 ± 2.99	2.31 ± 1.07	2.93 ± 1.44	1.23 ± 0.92
	mean	agent	4.52 ± 1.28	4.42 ± 0.59	2.71 ± 2.20	3.76 ± 2.22	1.64 ± 1.37	2.29 ± 2.09	1.00 ± 0.00
	(last 10)	client	4.41 ± 1.68	4.42 ± 0.75	2.71 ± 1.69	3.10 ± 1.66	1.53 ± 1.12	1.96 ± 1.57	1.00 ± 0.00
	median	agent	10.00	10.00	1.00	10.00	1.00	1.00	1.00
		client	10.00	10.00	1.00	1.00	1.00	1.00	1.00
	median	agent	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	(last 10)	client	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table A6: PD experiments with client strategy: (2t) and discount $\gamma = 0.9$

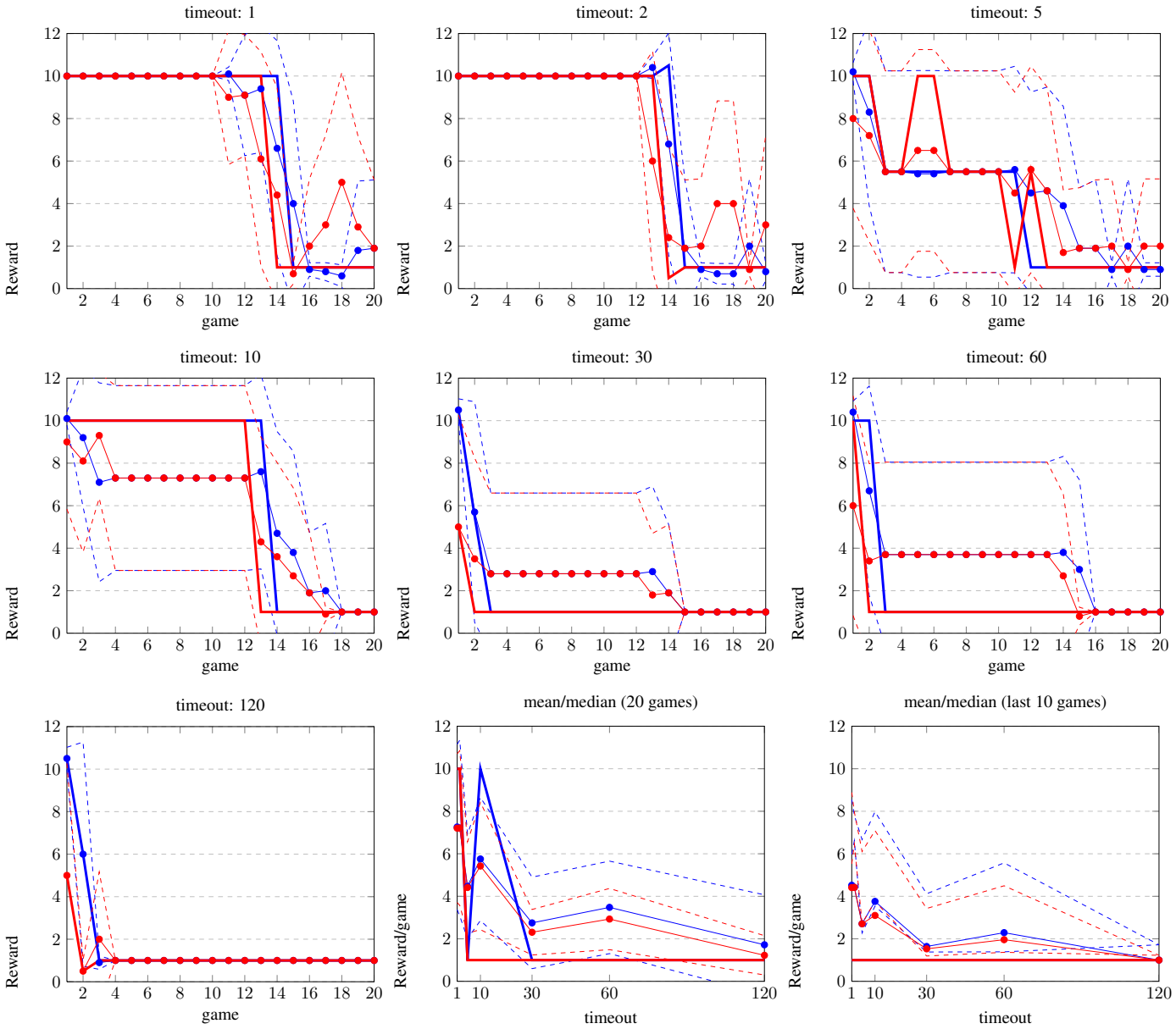

Figure A6: PD experiments with client strategy: (2t) and discount $\gamma = 0.9$ Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			timeout						
			1	2	5	10	30	60	120
0.90	mean	agent	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	8.93 ± 1.30	9.50 ± 0.92	9.35 ± 0.94	9.21 ± 1.34
		client	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	8.38 ± 1.62	8.56 ± 1.56	8.09 ± 1.94	7.46 ± 2.40
	mean	agent	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	7.89 ± 2.74	8.95 ± 1.24	8.64 ± 1.86	8.36 ± 1.74
	(last 10)	client	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	7.34 ± 3.74	7.63 ± 3.52	6.77 ± 4.01	5.61 ± 3.91
	median	agent	10.00	10.00	10.00	10.00	10.00	10.00	10.00
		client	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	median	agent	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	(last 10)	client	10.00	10.00	10.00	10.00	10.00	10.00	10.00

Table A7: PD experiments with client strategy: (1.0) and discount $\gamma = 0.9$

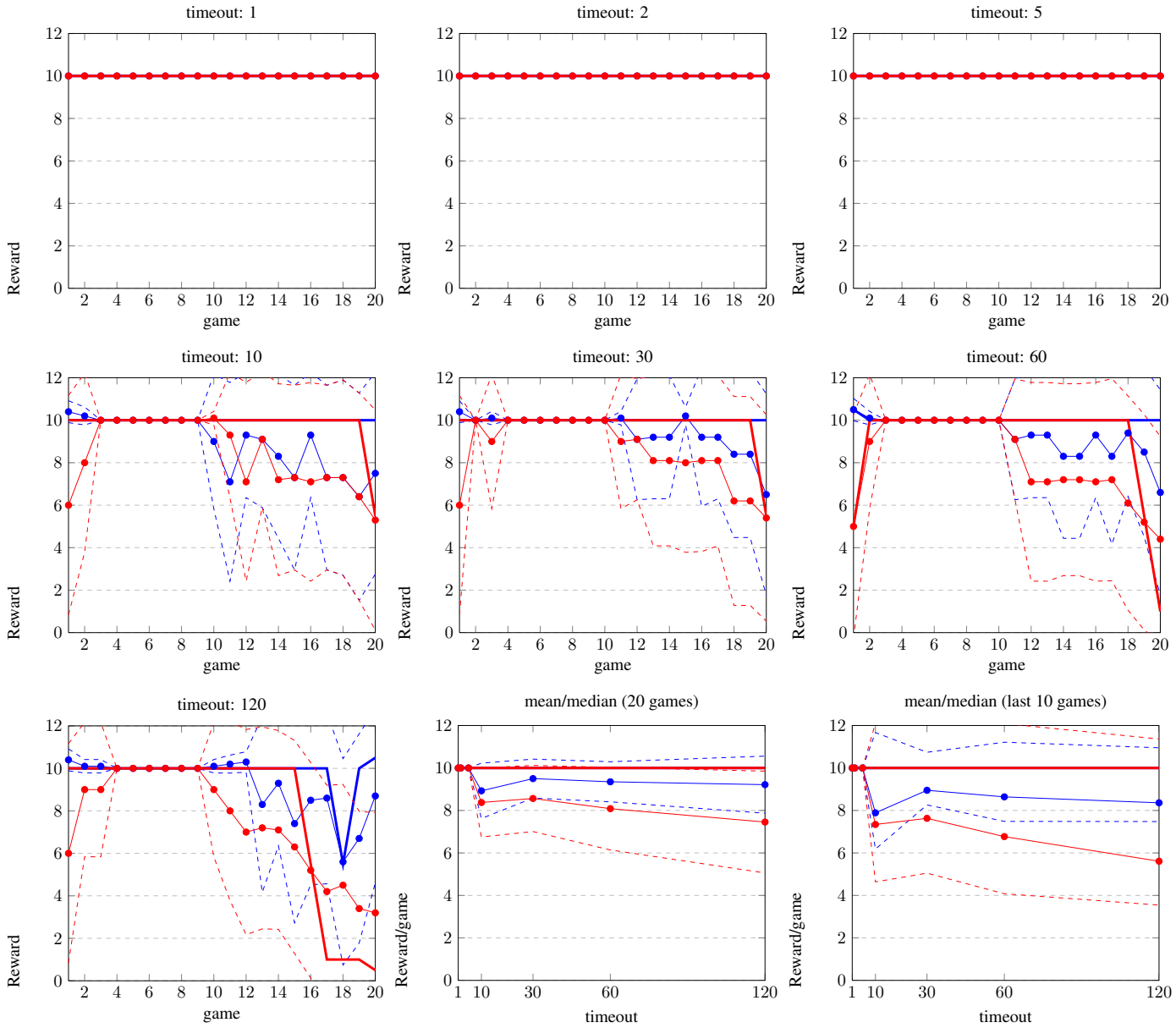

Figure A7: PD experiments with client strategy: (1.0) and discount $\gamma = 0.9$ Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			timeout						
			1	2	5	10	30	60	120
0.90	mean	agent	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	8.54 ± 1.15	8.91 ± 0.99	4.14 ± 2.38	3.14 ± 1.96
		client	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	8.27 ± 1.35	8.47 ± 1.11	3.71 ± 2.36	2.87 ± 2.13
	mean	agent	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	7.60 ± 3.32	8.10 ± 3.75	2.08 ± 2.84	1.65 ± 2.09
	(last 10)	client	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	7.27 ± 3.76	8.21 ± 3.80	2.08 ± 2.84	1.43 ± 1.06
	median	agent	10.00	10.00	10.00	10.00	10.00	1.00	1.00
		client	10.00	10.00	10.00	10.00	10.00	1.00	1.00
	median	agent	10.00	10.00	10.00	10.00	10.00	1.00	1.00
	(last 10)	client	10.00	10.00	10.00	10.00	10.00	1.00	1.00

Table A8: PD experiments with client strategy: (same) and discount $\gamma = 0.9$

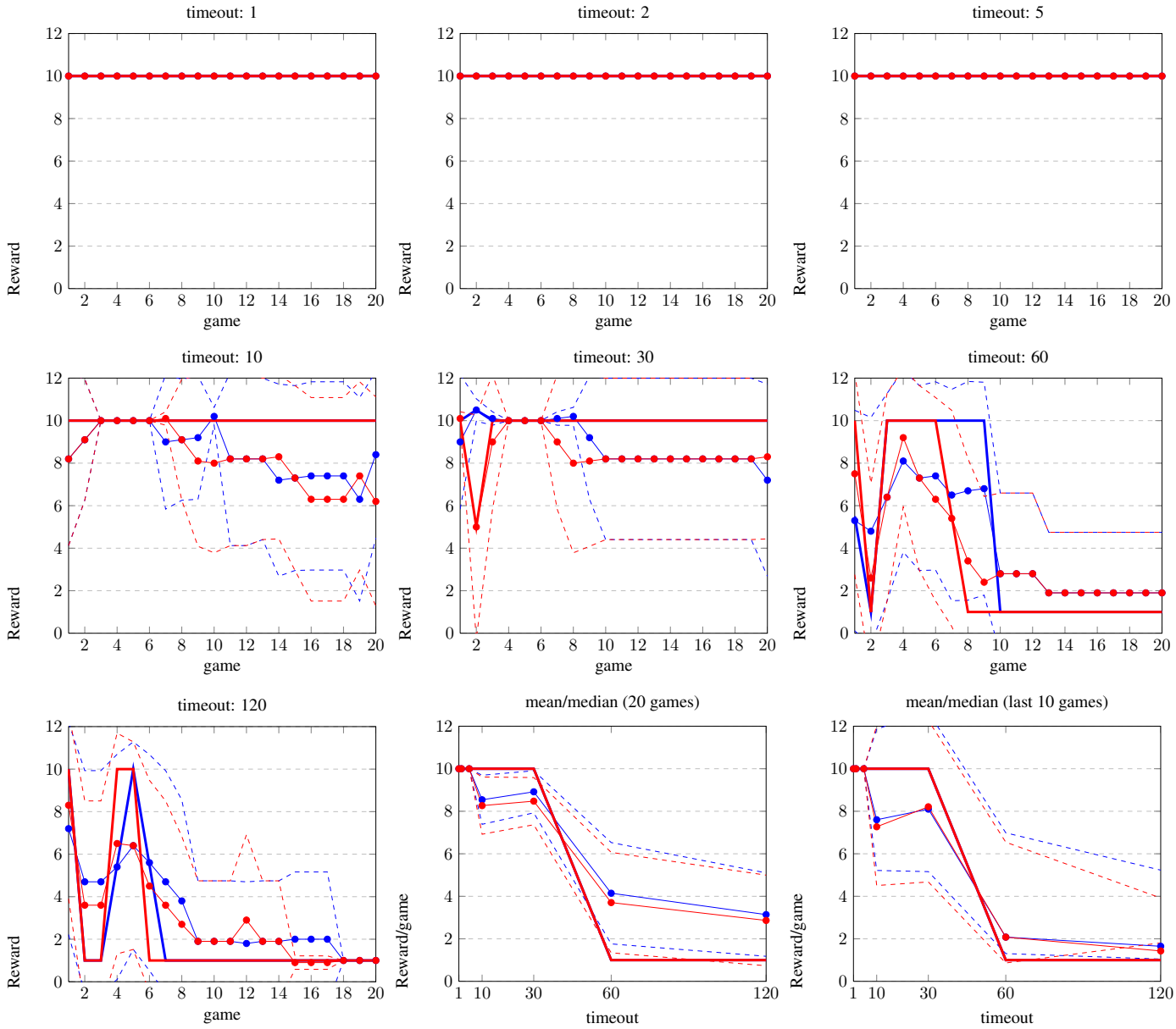

Figure A8: PD experiments with client strategy: (same) and discount $\gamma = 0.9$ Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			timeout						
			1	2	5	10	30	60	120
0.99	mean	agent	10.33 ± 0.15	10.33 ± 0.18	10.06 ± 0.14	10.07 ± 0.07	10.07 ± 0.06	10.02 ± 0.04	10.47 ± 0.14
		client	6.70 ± 1.49	6.70 ± 1.78	9.40 ± 1.35	9.30 ± 0.66	9.25 ± 0.64	9.80 ± 0.41	5.25 ± 1.37
	mean	agent	10.39 ± 0.12	10.43 ± 0.19	10.12 ± 0.10	10.10 ± 0.25	10.09 ± 0.19	10.04 ± 0.13	10.54 ± 0.38
	(last 10)	client	6.10 ± 1.20	5.70 ± 1.95	8.80 ± 1.03	9.00 ± 2.49	9.10 ± 1.91	9.60 ± 1.26	4.60 ± 3.84
	median	agent	10.00	10.00	10.00	10.00	10.00	10.00	10.00
		client	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	median	agent	10.00	10.00	10.00	10.00	10.00	10.00	11.00
	(last 10)	client	10.00	10.00	10.00	10.00	10.00	10.00	0.00

Table A9: PD experiments with client strategy: (co) and discount $\gamma = 0.99$

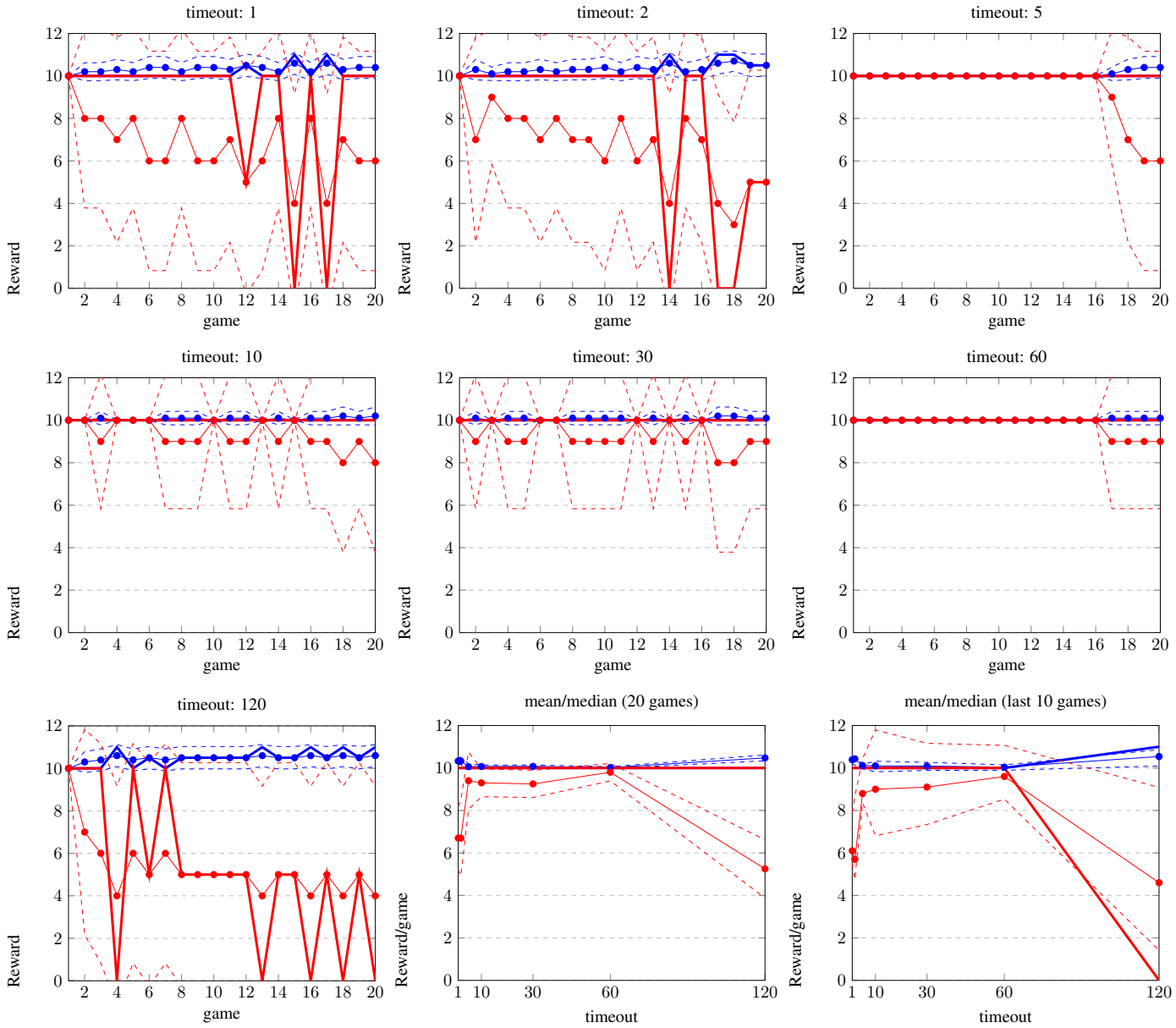

Figure A9: PD experiments with client strategy: (co) and discount $\gamma = 0.99$ Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			timeout						
			1	2	5	10	30	60	120
0.99	mean	agent	0.59 ± 0.26	0.55 ± 0.21	0.47 ± 0.20	0.58 ± 0.22	0.67 ± 0.21	0.64 ± 0.26	0.77 ± 0.22
		client	5.10 ± 2.61	5.50 ± 2.12	6.35 ± 1.98	5.20 ± 2.17	4.35 ± 2.11	4.65 ± 2.60	3.30 ± 2.23
	mean	agent	0.73 ± 0.19	0.64 ± 0.19	0.50 ± 0.20	0.66 ± 0.15	0.72 ± 0.09	0.76 ± 0.11	0.87 ± 0.15
	(last 10)	client	3.70 ± 1.89	4.60 ± 1.90	6.00 ± 2.00	4.40 ± 1.51	3.80 ± 0.92	3.40 ± 1.07	2.30 ± 1.49
	median	agent	1.00	1.00	0.00	1.00	1.00	1.00	1.00
		client	1.00	1.00	11.00	1.00	1.00	1.00	1.00
	median	agent	1.00	1.00	0.50	1.00	1.00	1.00	1.00
	(last 10)	client	1.00	1.00	6.00	1.00	1.00	1.00	1.00

Table A10: PD experiments with client strategy: (de) and discount $\gamma = 0.99$

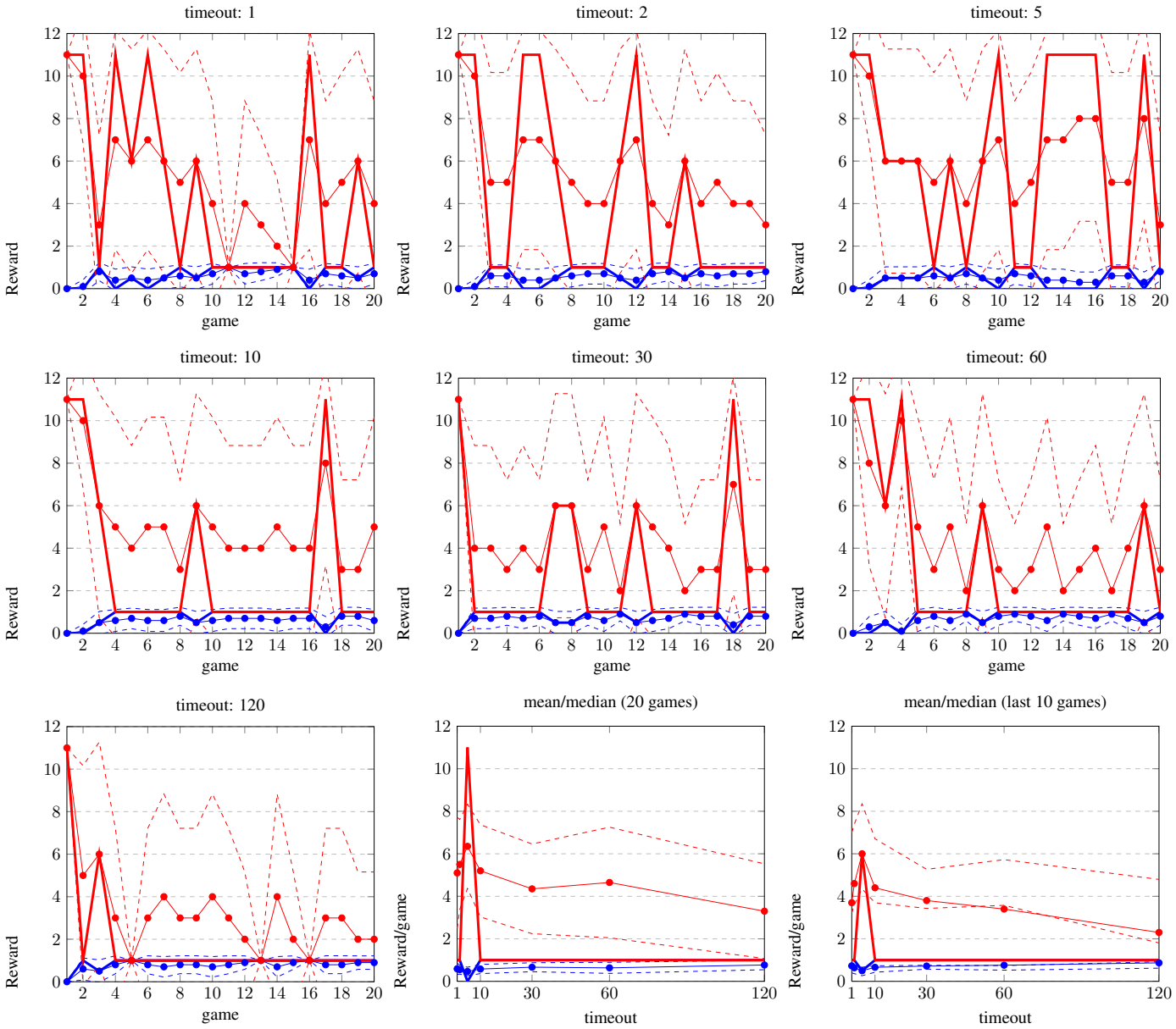

Figure A10: PD experiments with client strategy: (de) and discount $\gamma = 0.99$ Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			timeout						
			1	2	5	10	30	60	120
0.99	mean	agent	1.54 ± 2.95	1.52 ± 2.97	1.38 ± 2.97	1.48 ± 2.93	1.57 ± 2.96	1.60 ± 2.91	1.68 ± 2.90
		client	5.45 ± 2.63	5.65 ± 2.25	7.10 ± 2.07	6.10 ± 2.97	5.15 ± 2.28	4.85 ± 2.54	4.10 ± 2.22
	mean	agent	0.72 ± 0.19	0.69 ± 0.12	0.51 ± 0.19	0.64 ± 0.16	0.71 ± 0.19	0.78 ± 0.14	0.81 ± 0.11
	(last 10)	client	3.80 ± 1.87	4.10 ± 1.20	5.90 ± 1.91	4.60 ± 1.58	3.90 ± 1.85	3.20 ± 1.40	2.90 ± 1.10
	median	agent	1.00	1.00	0.00	1.00	1.00	1.00	1.00
		client	1.00	1.00	11.00	10.00	1.00	1.00	1.00
	median	agent	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	(last 10)	client	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table A11: PD experiments with client strategy: (to) and discount $\gamma = 0.99$

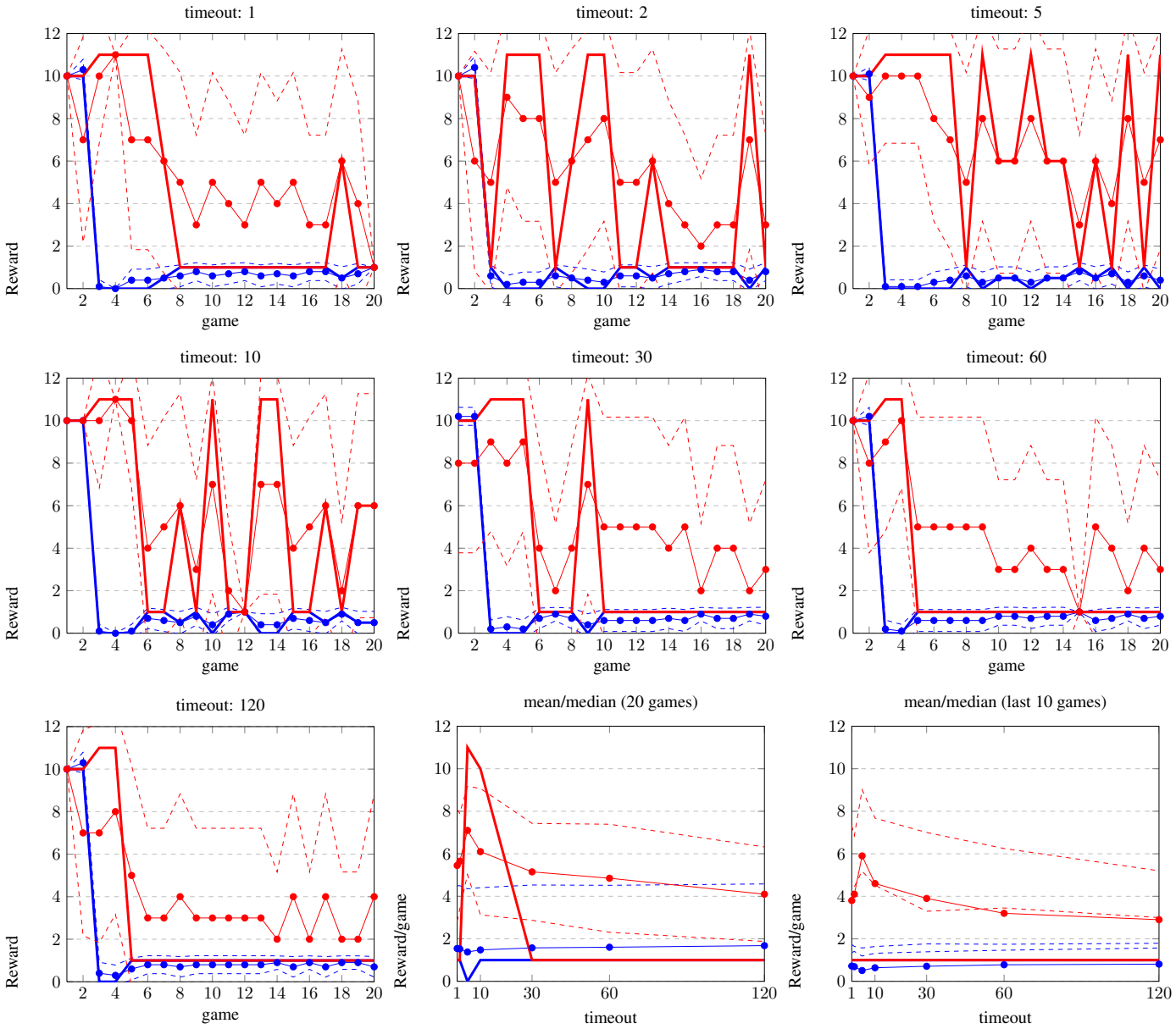

Figure A11: PD experiments with client strategy: (to) and discount $\gamma = 0.99$ Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			timeout						
			1	2	5	10	30	60	120
0.99	mean	agent	7.29 ± 1.87	7.01 ± 1.73	9.16 ± 1.32	10.00 ± 0.00	8.38 ± 0.92	7.33 ± 1.42	7.33 ± 1.17
		client	6.96 ± 1.83	6.74 ± 1.58	8.95 ± 1.46	10.00 ± 0.00	8.33 ± 0.85	7.05 ± 1.30	7.05 ± 1.02
	mean	agent	5.89 ± 1.26	6.41 ± 1.56	8.69 ± 1.41	10.00 ± 0.00	8.28 ± 2.82	7.04 ± 3.45	6.87 ± 3.75
	(last 10)	client	5.56 ± 0.99	6.30 ± 1.29	8.25 ± 1.64	10.00 ± 0.00	8.39 ± 2.59	6.93 ± 3.48	6.65 ± 3.83
	median	agent	10.00	10.00	10.00	10.00	10.00	10.00	10.00
		client	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	median	agent	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	(last 10)	client	10.00	10.00	10.00	10.00	10.00	10.00	10.00

Table A12: PD experiments with client strategy: (tt) and discount $\gamma = 0.99$

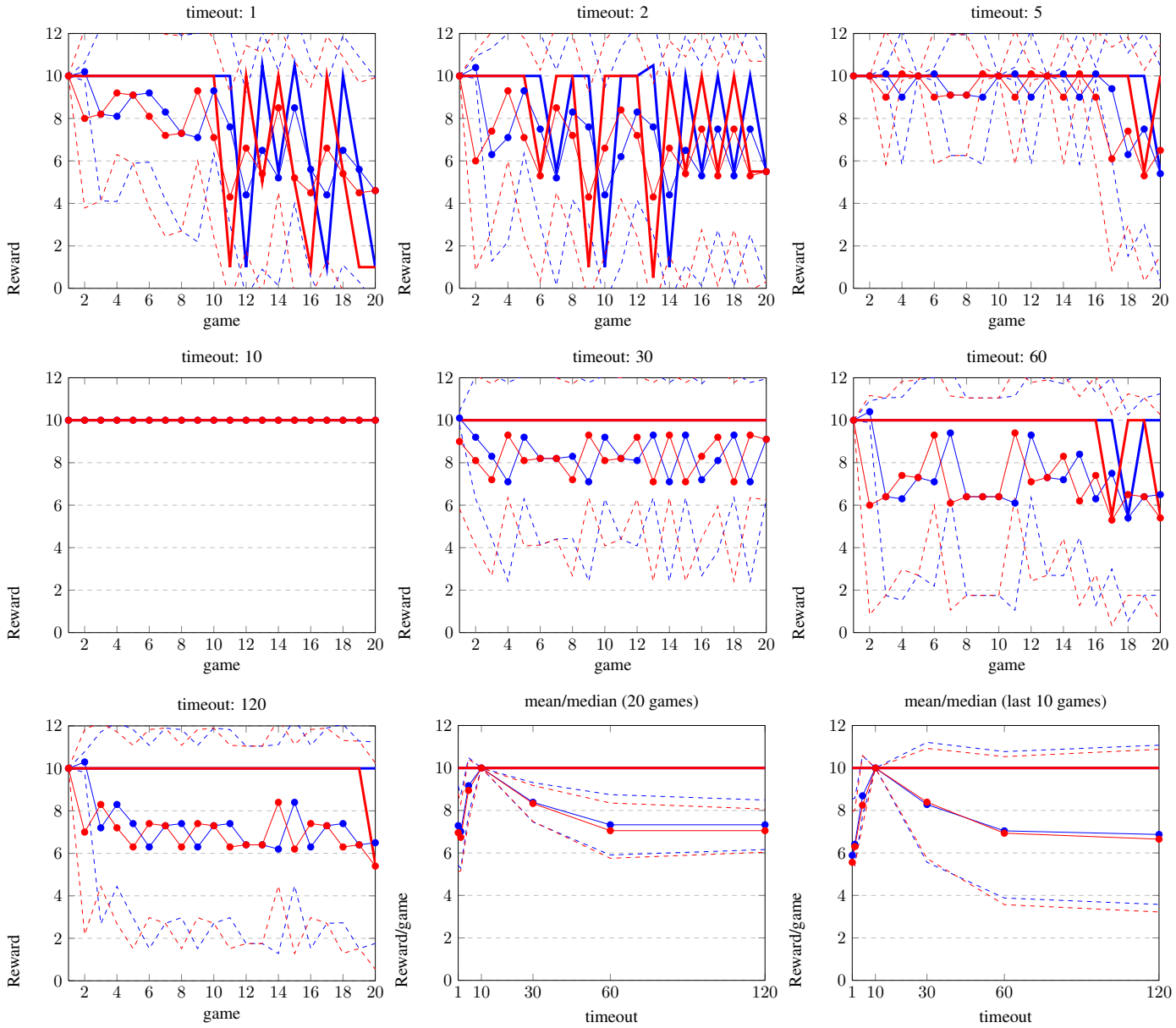

Figure A12: PD experiments with client strategy: (tt) and discount $\gamma = 0.99$ Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			timeout						
			1	2	5	10	30	60	120
0.99	mean	agent	9.47 ± 1.15	9.66 ± 0.75	9.90 ± 0.42	9.96 ± 0.20	9.97 ± 0.16	9.25 ± 0.80	7.28 ± 1.68
		client	6.83 ± 2.15	8.45 ± 1.45	9.02 ± 1.68	9.46 ± 1.25	9.76 ± 0.70	8.59 ± 0.77	6.01 ± 1.40
	mean	agent	9.17 ± 1.08	9.35 ± 1.18	9.87 ± 0.55	9.91 ± 0.54	9.95 ± 0.23	9.17 ± 1.91	6.57 ± 3.75
	(last 10)	client	5.43 ± 1.08	7.59 ± 1.50	8.33 ± 1.44	8.92 ± 1.32	9.51 ± 0.94	8.40 ± 2.42	5.69 ± 4.00
	median	agent	10.00	10.00	10.00	10.00	10.00	10.00	10.00
		client	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	median	agent	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	(last 10)	client	10.00	10.00	10.00	10.00	10.00	10.00	10.00

Table A13: PD experiments with client strategy: (t2) and discount $\gamma = 0.99$

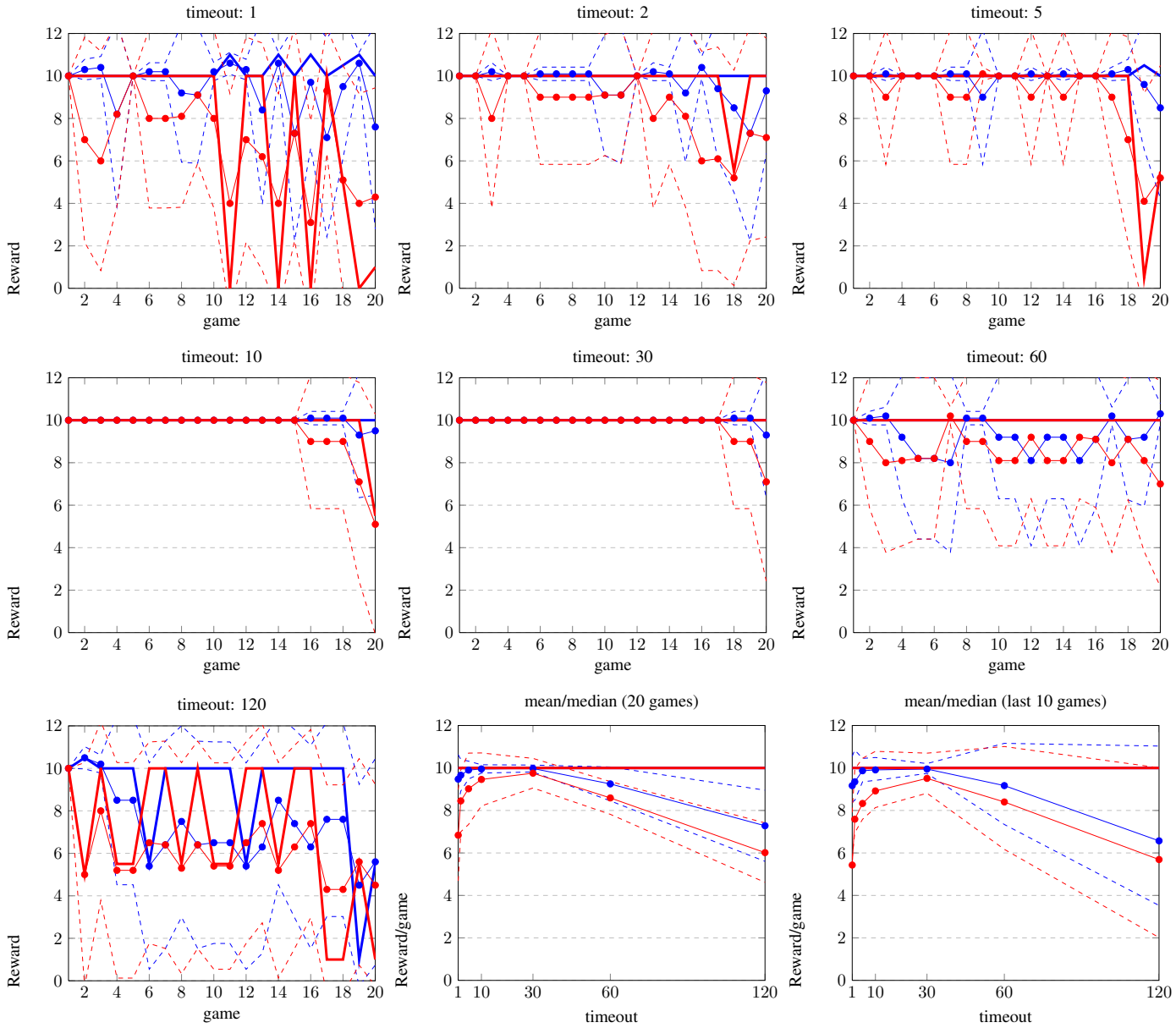

Figure A13: PD experiments with client strategy: (t2) and discount $\gamma = 0.99$ Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			timeout						
			1	2	5	10	30	60	120
0.99	mean	agent	5.25 ± 2.75	7.42 ± 2.35	8.72 ± 1.41	9.90 ± 0.31	8.32 ± 0.90	8.45 ± 0.87	7.63 ± 0.91
		client	7.01 ± 1.80	8.08 ± 1.96	8.89 ± 1.27	9.96 ± 0.23	8.93 ± 0.91	8.67 ± 1.20	7.91 ± 0.84
	mean	agent	2.90 ± 2.20	5.84 ± 2.83	8.19 ± 2.62	9.81 ± 0.60	7.86 ± 3.76	8.00 ± 3.84	7.33 ± 4.31
	(last 10)	client	5.76 ± 1.04	7.05 ± 1.51	8.30 ± 1.53	9.92 ± 0.25	8.63 ± 2.22	8.22 ± 2.61	7.99 ± 3.25
	median	agent	1.00	10.00	10.00	10.00	10.00	10.00	10.00
		client	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	median	agent	1.00	10.00	10.00	10.00	10.00	10.00	10.00
	(last 10)	client	5.50	10.00	10.00	10.00	10.00	10.00	10.00

Table A14: PD experiments with client strategy: (2t) and discount $\gamma = 0.99$

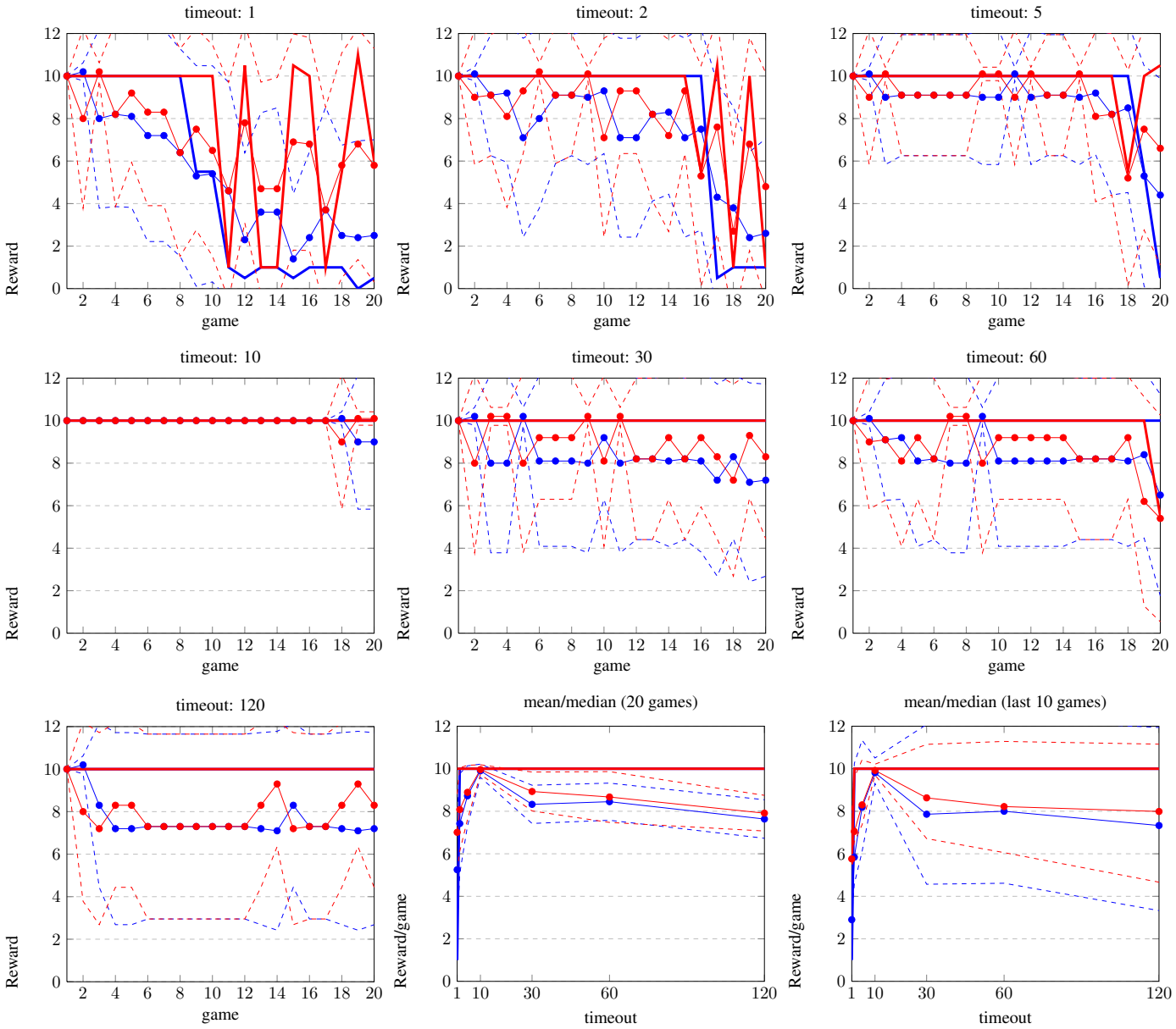

Figure A14: PD experiments with client strategy: (2t) and discount $\gamma = 0.99$ Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			timeout						
			1	2	5	10	30	60	120
0.99	mean	agent	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	9.41 ± 1.17	9.79 ± 0.43	10.00 ± 0.00
		client	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	8.97 ± 1.25	9.68 ± 0.48	10.00 ± 0.00
	mean	agent	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	8.79 ± 1.60	9.67 ± 1.04	10.00 ± 0.00
	(last 10)	client	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	8.24 ± 2.29	9.34 ± 2.09	10.00 ± 0.00
	median	agent	10.00	10.00	10.00	10.00	10.00	10.00	10.00
		client	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	median	agent	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	(last 10)	client	10.00	10.00	10.00	10.00	10.00	10.00	10.00

Table A15: PD experiments with client strategy: (1.0) and discount $\gamma = 0.99$

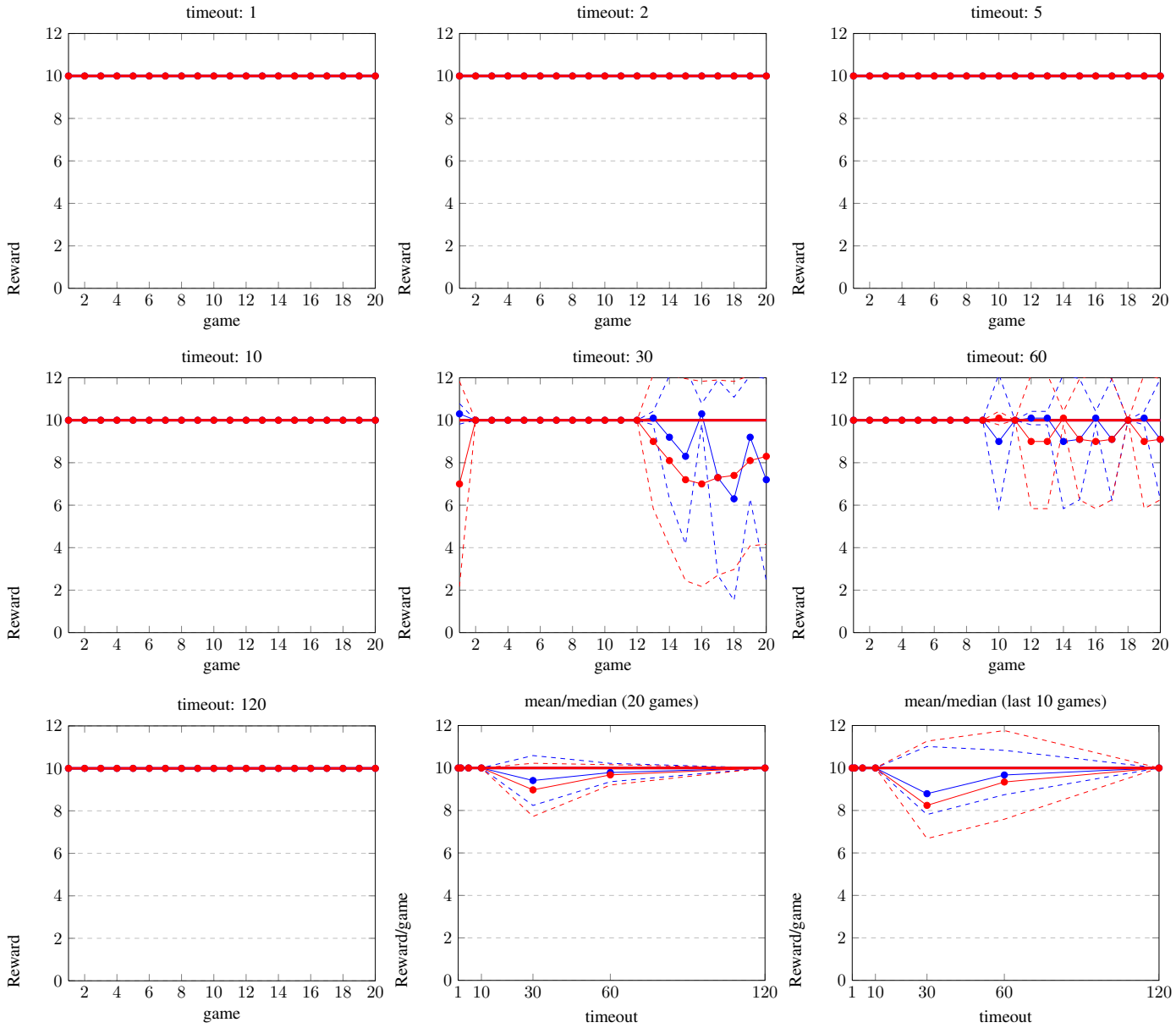

Figure A15: PD experiments with client strategy: (1.0) and discount $\gamma = 0.99$ Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			timeout						
			1	2	5	10	30	60	120
0.99	mean	agent	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	8.15 ± 1.55	9.40 ± 0.46
		client	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	8.21 ± 1.50	9.52 ± 0.48
	mean	agent	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	6.91 ± 3.38	9.26 ± 2.34
	(last 10)	client	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	7.24 ± 3.08	9.48 ± 1.64
	median	agent	10.00	10.00	10.00	10.00	10.00	10.00	10.00
		client	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	median	agent	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	(last 10)	client	10.00	10.00	10.00	10.00	10.00	10.00	10.00

Table A16: PD experiments with client strategy: (same) and discount $\gamma = 0.99$

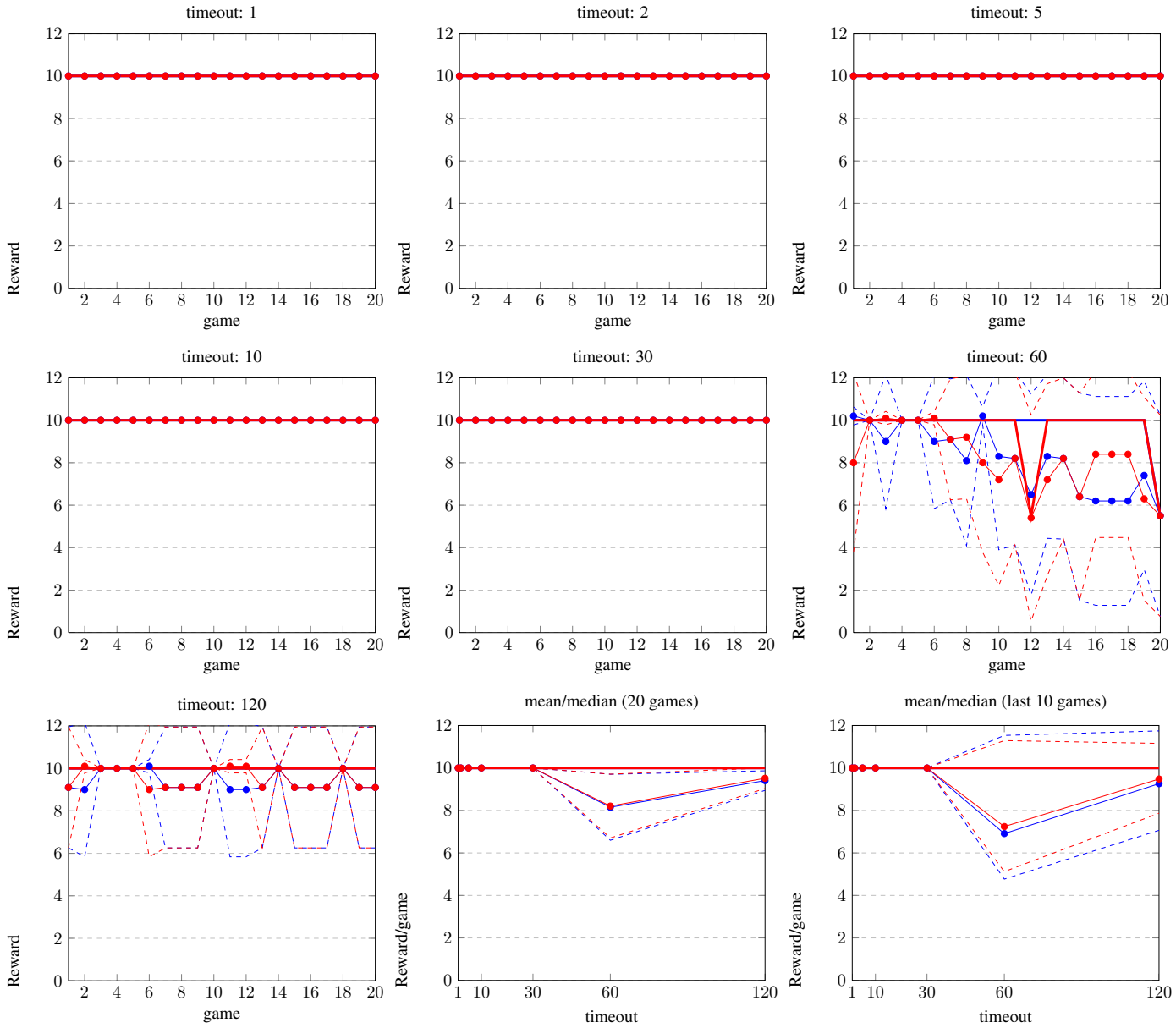

Figure A16: PD experiments with client strategy: (same) and discount $\gamma = 0.99$ Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			σ_b						
			0.01	0.05	0.10	0.50	1.00	2.00	5.00
0.99	mean	agent	10.64 ± 0.27	10.77 ± 0.22	10.23 ± 0.10	10.02 ± 0.05	10.04 ± 0.08	10.00 ± 0.00	10.00 ± 0.00
		client	3.60 ± 2.74	2.35 ± 2.18	7.70 ± 1.03	9.85 ± 0.49	9.60 ± 0.75	10.00 ± 0.00	10.00 ± 0.00
	mean	agent	10.75 ± 0.13	10.80 ± 0.11	10.28 ± 0.39	10.03 ± 0.07	10.08 ± 0.16	10.00 ± 0.00	10.00 ± 0.00
	(last 10)	client	2.50 ± 1.27	2.00 ± 1.05	7.20 ± 3.88	9.70 ± 0.67	9.20 ± 1.62	10.00 ± 0.00	10.00 ± 0.00
	median	agent	11.00	11.00	10.00	10.00	10.00	10.00	10.00
		client	0.00	0.00	10.00	10.00	10.00	10.00	10.00
	median	agent	11.00	11.00	10.00	10.00	10.00	10.00	10.00
	(last 10)	client	0.00	0.00	10.00	10.00	10.00	10.00	10.00

Table A17: PD experiments with client strategy: (co), timeout=120.0 and discount $\gamma = 0.99$.

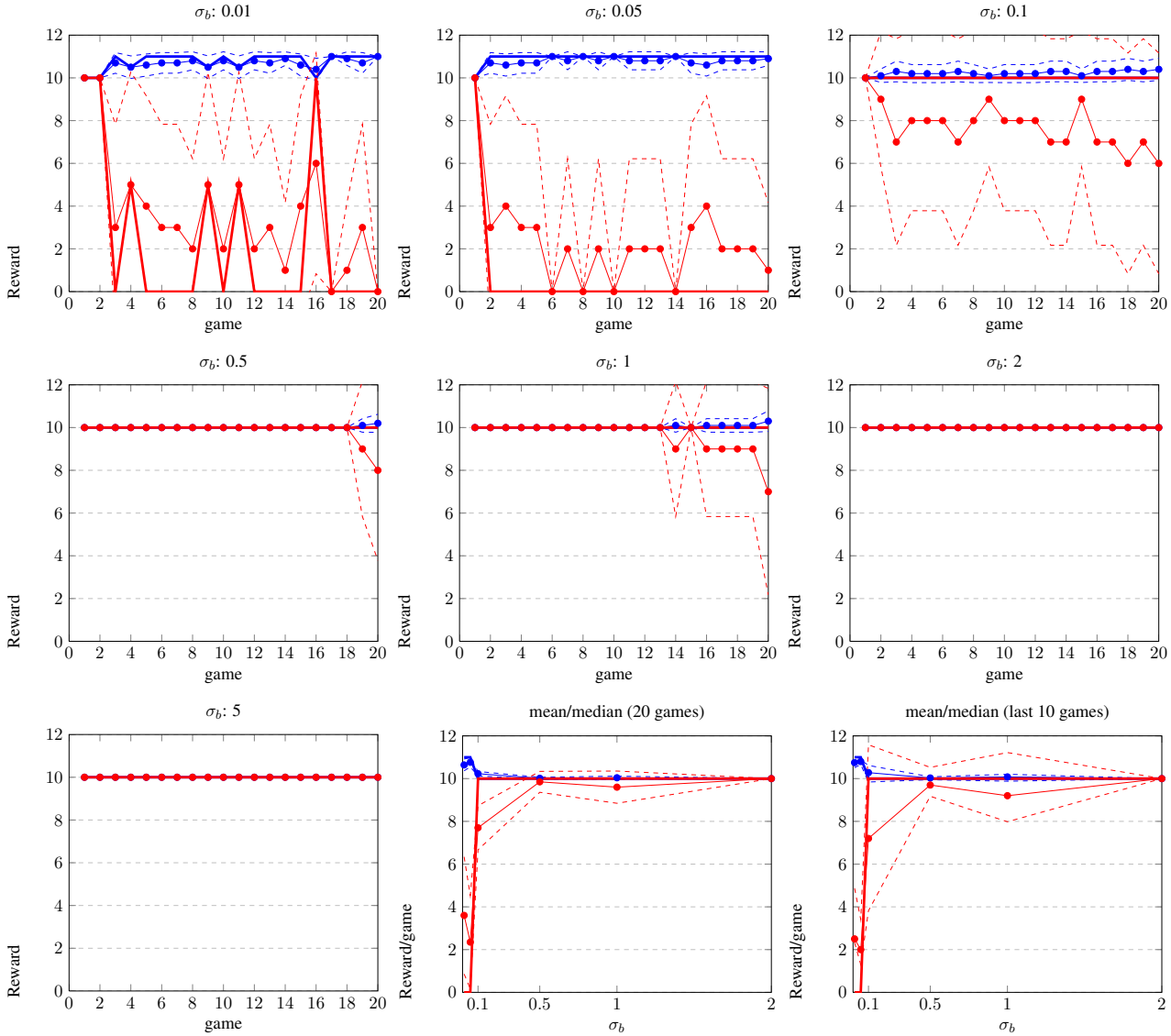

Figure A17: PD experiments with client strategy: (co), timeout=120.0 and discount $\gamma = 0.99$. Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			σ_b						
			0.01	0.05	0.10	0.50	1.00	2.00	5.00
0.99	mean	agent	0.71 ± 0.23	0.74 ± 0.31	0.78 ± 0.26	0.78 ± 0.24	0.73 ± 0.25	0.76 ± 0.27	0.73 ± 0.24
		client	3.85 ± 2.32	3.60 ± 3.10	3.25 ± 2.55	3.20 ± 2.38	3.65 ± 2.48	3.40 ± 2.74	3.70 ± 2.36
	mean	agent	0.81 ± 0.11	0.88 ± 0.06	0.87 ± 0.13	0.86 ± 0.13	0.83 ± 0.14	0.89 ± 0.12	0.79 ± 0.12
	(last 10)	client	2.90 ± 1.10	2.20 ± 0.63	2.30 ± 1.25	2.40 ± 1.35	2.70 ± 1.42	2.10 ± 1.20	3.10 ± 1.20
	median	agent	1.00	1.00	1.00	1.00	1.00	1.00	1.00
		client	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	median	agent	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	(last 10)	client	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table A18: PD experiments with client strategy: (de), timeout=120.0 and discount $\gamma = 0.99$.

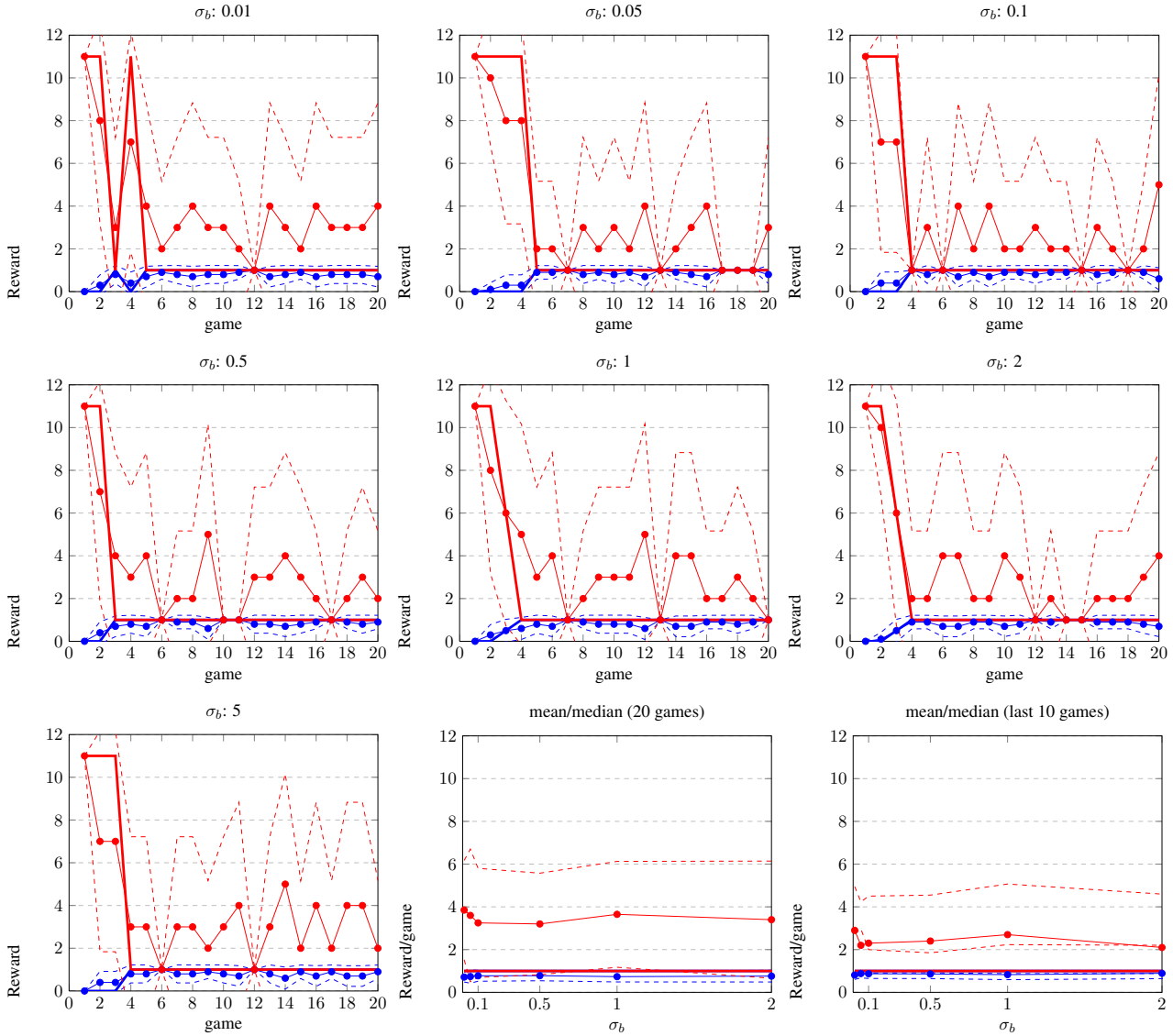

Figure A18: PD experiments with client strategy: (de), timeout=120.0 and discount $\gamma = 0.99$. Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			σ_b						
			0.01	0.05	0.10	0.50	1.00	2.00	5.00
0.99	mean	agent	1.69 ± 2.85	1.76 ± 2.93	1.70 ± 2.89	1.62 ± 2.88	1.52 ± 2.91	1.60 ± 2.89	1.60 ± 2.89
		client	4.05 ± 2.31	3.30 ± 2.32	3.95 ± 3.10	4.70 ± 3.74	5.65 ± 3.08	4.85 ± 3.38	4.85 ± 3.42
	mean	agent	0.77 ± 0.13	0.85 ± 0.12	0.88 ± 0.09	0.88 ± 0.11	0.72 ± 0.18	0.82 ± 0.16	0.82 ± 0.14
	(last 10)	client	3.30 ± 1.34	2.50 ± 1.18	2.20 ± 0.92	2.20 ± 1.14	3.80 ± 1.81	2.80 ± 1.62	2.80 ± 1.40
	median	agent	1.00	1.00	1.00	1.00	1.00	1.00	1.00
		client	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	median	agent	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	(last 10)	client	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table A19: PD experiments with client strategy: (to), timeout=120.0 and discount $\gamma = 0.99$.

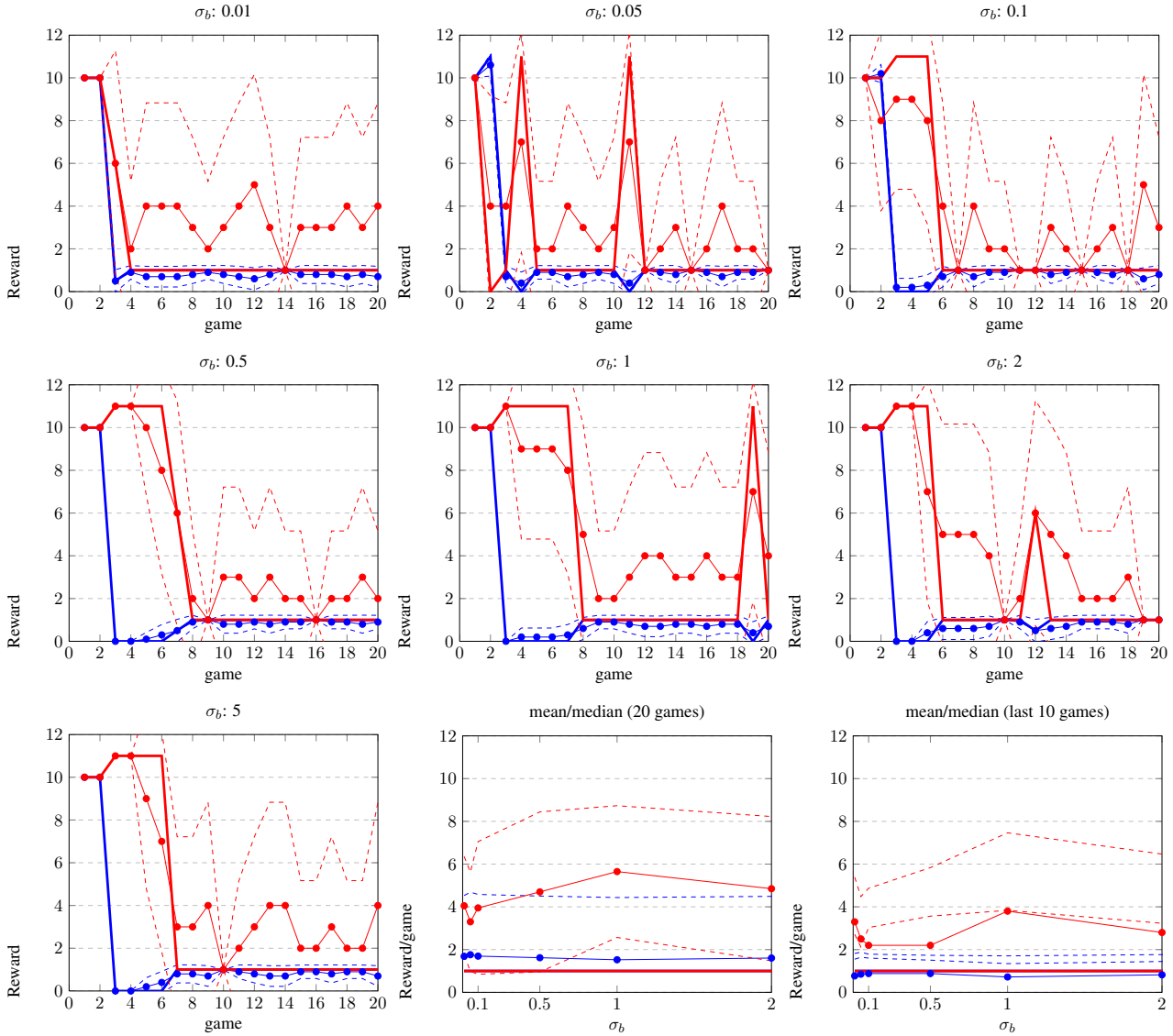

Figure A19: PD experiments with client strategy: (to), timeout=120.0 and discount $\gamma = 0.99$. Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			σ_b						
			0.01	0.05	0.10	0.50	1.00	2.00	5.00
0.99	mean	agent	4.67 ± 2.76	3.42 ± 2.83	5.93 ± 1.84	9.34 ± 0.50	9.93 ± 0.25	9.82 ± 0.40	9.78 ± 0.44
		client	4.34 ± 2.39	2.98 ± 2.25	5.66 ± 1.49	9.18 ± 0.68	9.76 ± 0.69	9.77 ± 0.43	9.72 ± 0.47
	mean	agent	3.87 ± 1.04	1.80 ± 0.69	4.96 ± 3.89	9.03 ± 2.83	9.85 ± 0.34	9.65 ± 1.11	9.65 ± 1.11
	(last 10)	client	3.98 ± 1.16	1.91 ± 0.62	4.96 ± 3.92	8.81 ± 2.82	9.52 ± 0.81	9.54 ± 1.45	9.54 ± 1.45
	median	agent	1.00	1.00	10.00	10.00	10.00	10.00	10.00
		client	1.00	1.00	10.00	10.00	10.00	10.00	10.00
	median	agent	1.00	1.00	1.00	10.00	10.00	10.00	10.00
	(last 10)	client	1.00	1.00	1.00	10.00	10.00	10.00	10.00

Table A20: PD experiments with client strategy: (tt), timeout=120.0 and discount $\gamma = 0.99$.

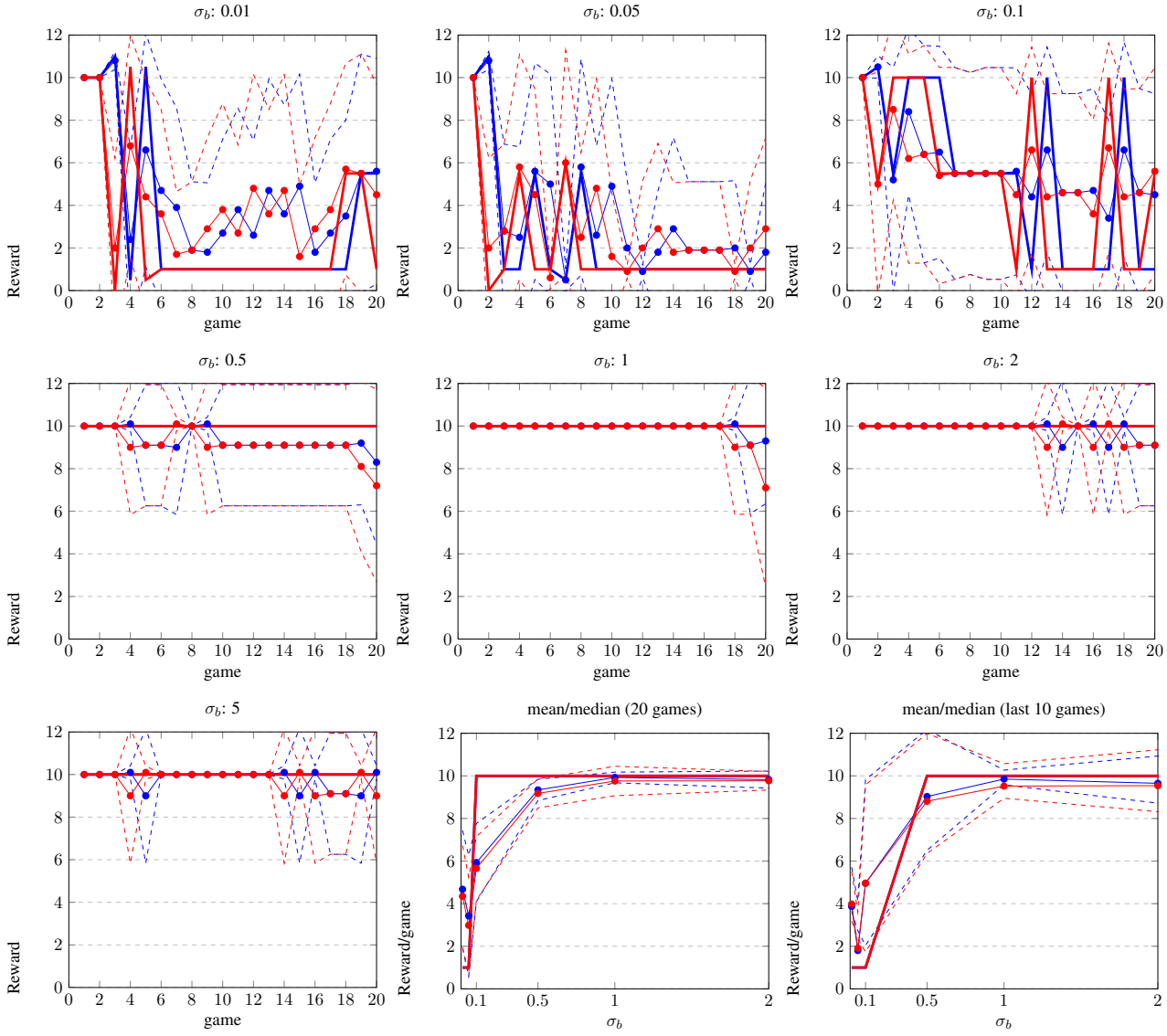

Figure A20: PD experiments with client strategy: (tt), timeout=120.0 and discount $\gamma = 0.99$. Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			σ_b						
			0.01	0.05	0.10	0.50	1.00	2.00	5.00
0.99	mean	agent	7.46 ± 2.33	4.83 ± 2.80	8.11 ± 1.18	9.90 ± 0.43	10.02 ± 0.05	10.02 ± 0.04	10.00 ± 0.00
		client	5.16 ± 2.58	2.75 ± 2.12	7.17 ± 1.03	9.46 ± 1.12	9.85 ± 0.49	9.85 ± 0.37	10.00 ± 0.00
	mean	agent	5.59 ± 2.47	3.73 ± 2.67	7.43 ± 3.42	9.81 ± 0.33	10.03 ± 0.05	10.03 ± 0.05	10.00 ± 0.00
	(last 10)	client	3.61 ± 1.67	2.41 ± 1.32	6.99 ± 3.92	8.93 ± 1.55	9.70 ± 0.48	9.70 ± 0.48	10.00 ± 0.00
	median	agent	10.00	1.00	10.00	10.00	10.00	10.00	10.00
		client	1.00	1.00	10.00	10.00	10.00	10.00	10.00
	median	agent	1.00	1.00	10.00	10.00	10.00	10.00	10.00
	(last 10)	client	1.00	1.00	10.00	10.00	10.00	10.00	10.00

Table A21: PD experiments with client strategy: (t2), timeout=120.0 and discount $\gamma = 0.99$.

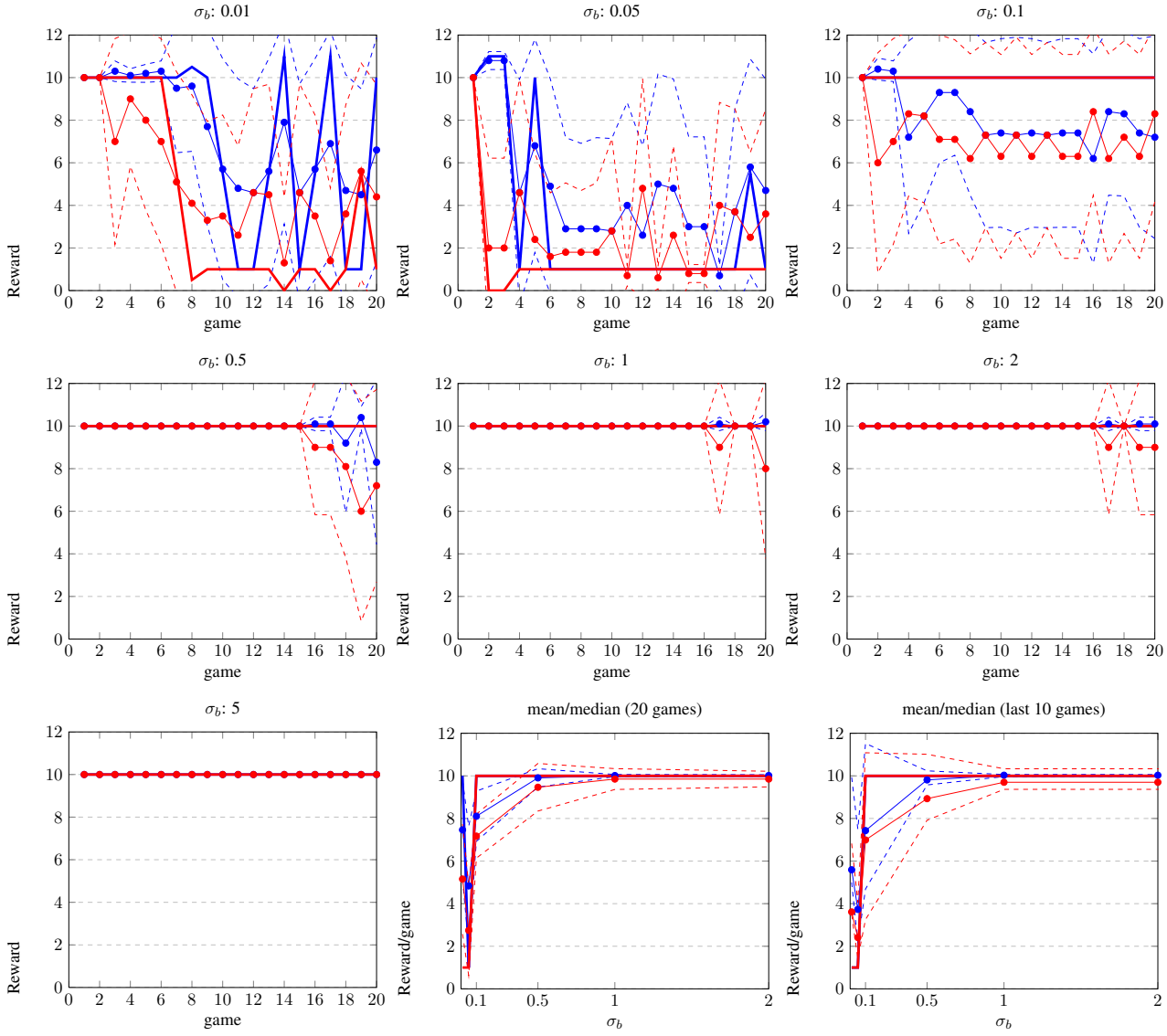

Figure A21: PD experiments with client strategy: (t2), timeout=120.0 and discount $\gamma = 0.99$. Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			σ_b						
			0.01	0.05	0.10	0.50	1.00	2.00	5.00
0.99	mean	agent	2.63 ± 3.39	2.69 ± 3.01	8.41 ± 0.83	9.76 ± 0.99	9.82 ± 0.40	9.78 ± 0.58	9.90 ± 0.31
		client	3.96 ± 2.56	3.85 ± 2.08	8.57 ± 1.21	9.48 ± 1.47	9.82 ± 0.40	9.72 ± 0.60	9.96 ± 0.23
	mean	agent	1.52 ± 0.57	1.12 ± 0.59	8.01 ± 3.62	9.51 ± 0.63	9.64 ± 1.14	9.56 ± 0.96	9.81 ± 0.60
	(last 10)	client	2.73 ± 1.17	2.77 ± 1.35	8.12 ± 2.66	8.96 ± 0.82	9.64 ± 1.14	9.45 ± 1.16	9.92 ± 0.25
	median	agent	1.00	1.00	10.00	10.00	10.00	10.00	10.00
		client	1.00	1.00	10.00	10.00	10.00	10.00	10.00
	median	agent	1.00	1.00	10.00	10.00	10.00	10.00	10.00
	(last 10)	client	1.00	1.00	10.00	10.00	10.00	10.00	10.00

Table A22: PD experiments with client strategy: (2t), timeout=120.0 and discount $\gamma = 0.99$.

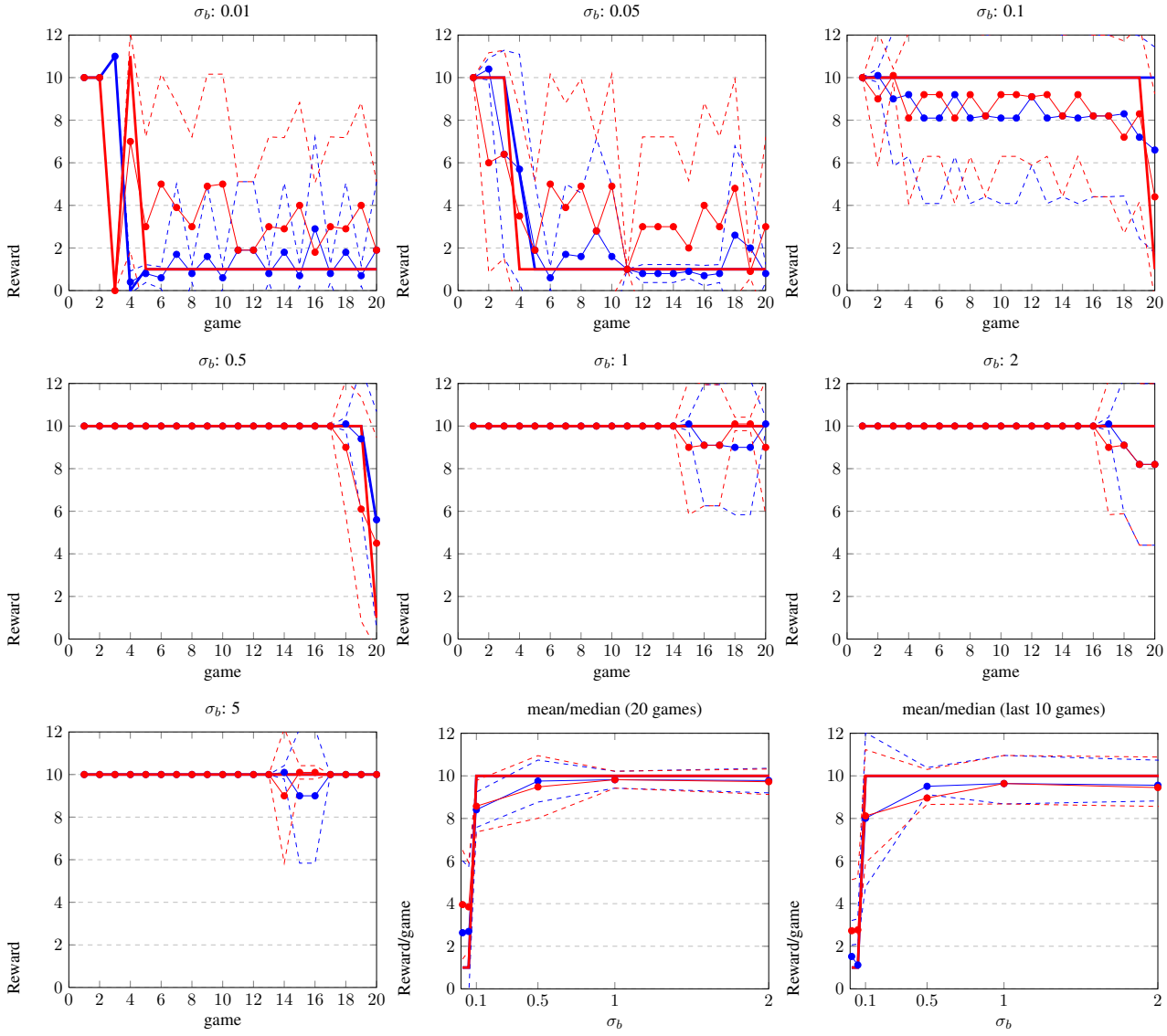

Figure A22: PD experiments with client strategy: (2t), timeout=120.0 and discount $\gamma = 0.99$. Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			σ_b						
			0.01	0.05	0.10	0.50	1.00	2.00	5.00
0.99	mean	agent	9.71 ± 0.51	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00
		client	9.38 ± 0.51	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00
	mean	agent	9.36 ± 2.02	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00
	(last 10)	client	9.47 ± 1.68	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00
	median	agent	10.00	10.00	10.00	10.00	10.00	10.00	10.00
		client	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	median	agent	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	(last 10)	client	10.00	10.00	10.00	10.00	10.00	10.00	10.00

Table A23: PD experiments with client strategy: (1.0), timeout=120.0 and discount $\gamma = 0.99$.

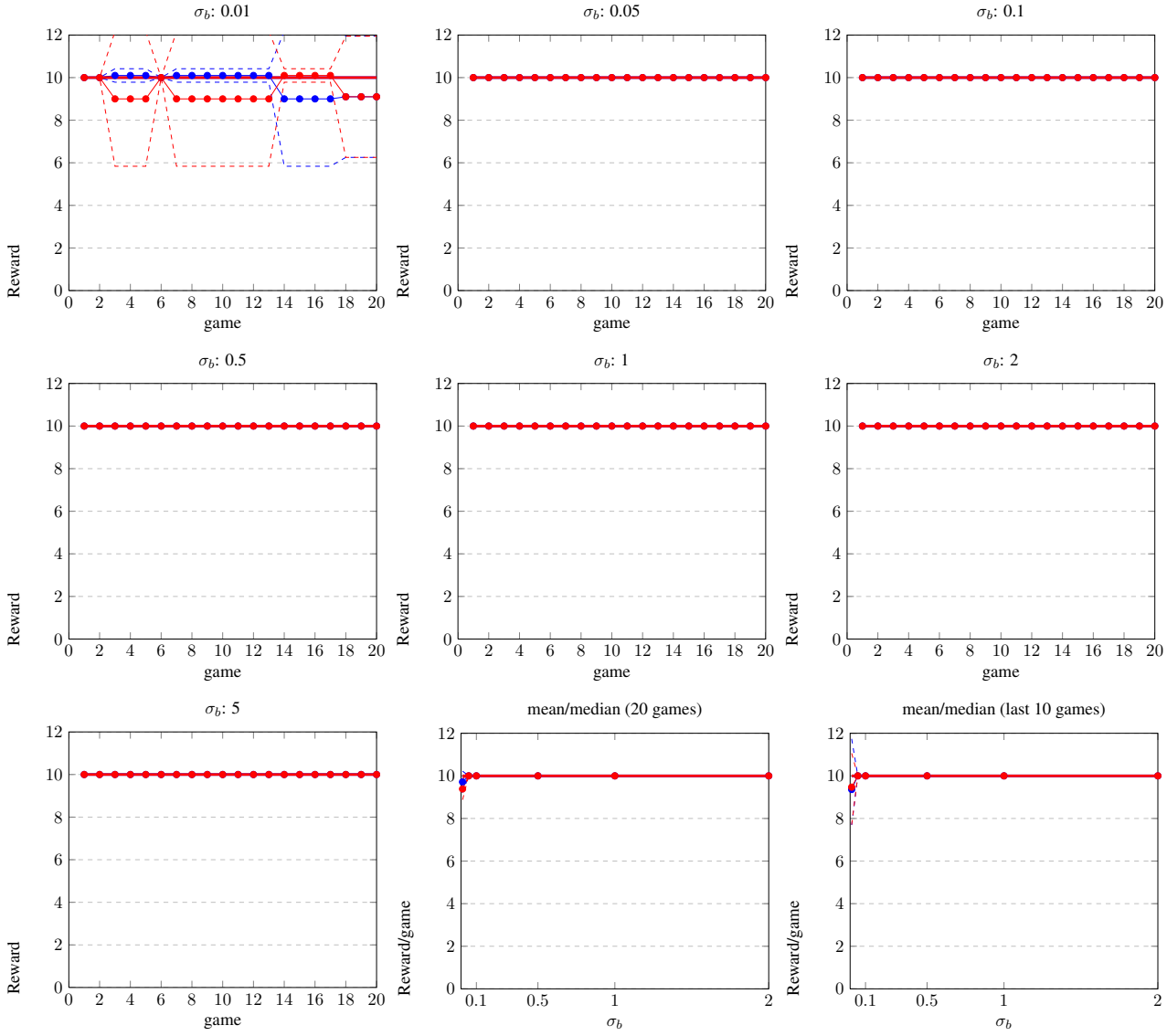

Figure A23: PD experiments with client strategy: (1.0), timeout=120.0 and discount $\gamma = 0.99$. Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.

γ			σ_b						
			0.01	0.05	0.10	0.50	1.00	2.00	5.00
0.99	mean	agent	10.00 ± 0.00	9.11 ± 0.80	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.01 ± 0.02	9.99 ± 0.22
		client	10.00 ± 0.00	9.00 ± 0.79	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	9.95 ± 0.22	9.61 ± 0.59
	mean	agent	10.00 ± 0.00	8.68 ± 2.79	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	9.96 ± 0.16
	(last 10)	client	10.00 ± 0.00	8.35 ± 3.51	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	10.00 ± 0.00	9.41 ± 1.55
	median	agent	10.00	10.00	10.00	10.00	10.00	10.00	10.00
		client	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	median	agent	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	(last 10)	client	10.00	10.00	10.00	10.00	10.00	10.00	10.00

Table A24: PD experiments with client strategy: (same), timeout=120.0 and discount $\gamma = 0.99$.

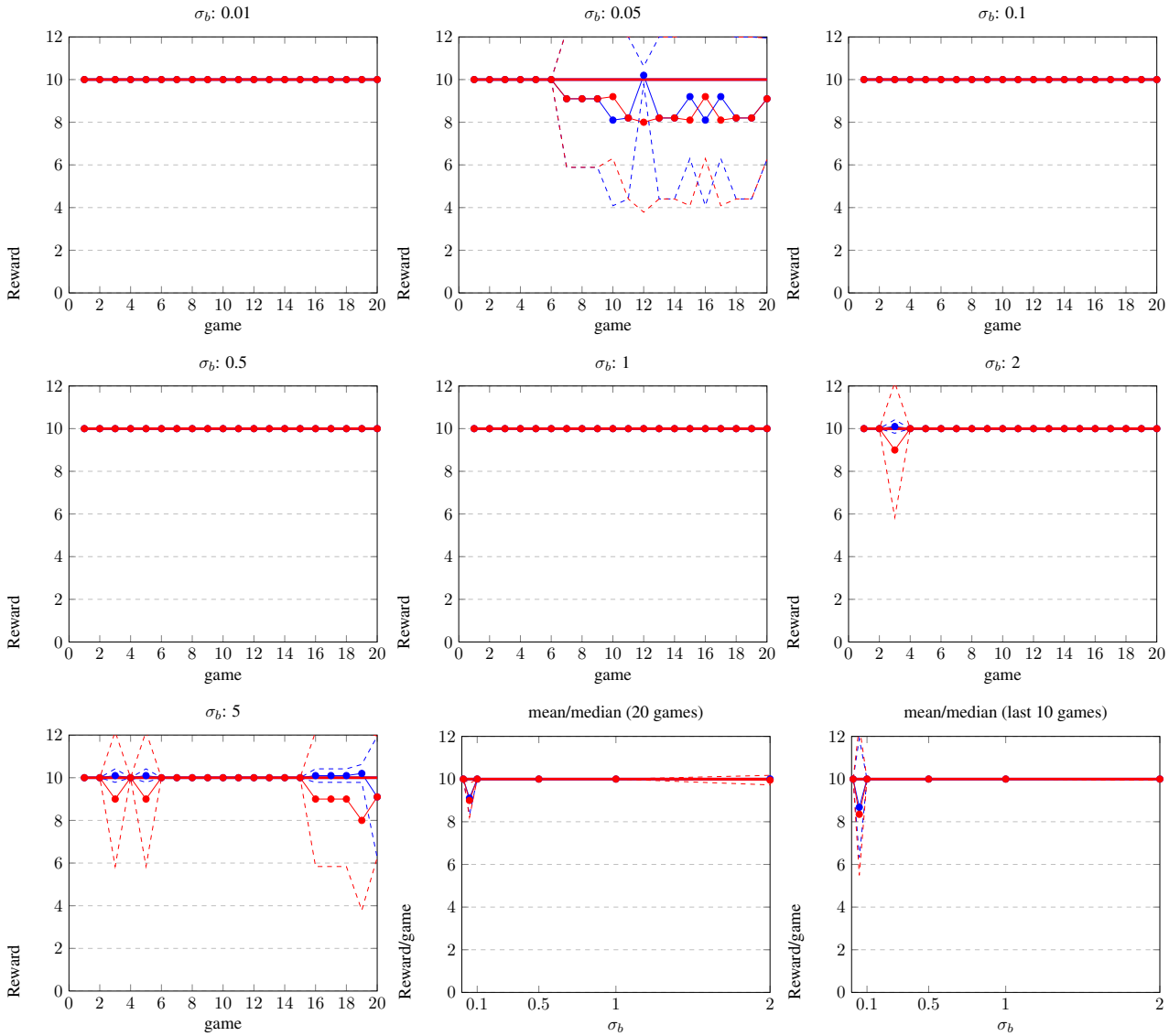

Figure A24: PD experiments with client strategy: (same), timeout=120.0 and discount $\gamma = 0.99$. Red=client, Blue=agent, dashed=std.dev. solid (thin, with markers): mean, solid (thick): median.