Distributed Relational Temporal Difference Learning

Qiangfeng Peter Lau[♠], Mong Li Lee[†] and Wynne Hsu[§]
Department of Computer Science
National University of Singapore
13 Computing Drive, Singapore 117417, Republic of Singapore
{plau[♠],leeml[†],whsu[§]}@comp.nus.edu.sg

ABSTRACT

Relational representations have great potential for rapidly generalizing learned knowledge in large Markov decision processes such as multi-agent problems. In this work, we introduce relational temporal difference learning for the distributed case where the communication links among agents are dynamic. Thus no critical components of the system should reside in any one agent. Relational generalization among agents' learning is achieved through the use of partially bound relational features and a message passing scheme. We further describe how the proposed concepts can be applied to distributed reinforcement learning methods that use value functions. Experiments were conducted on soccer and real-time strategy game domains with dynamic communication. Results show that our methods improve goal achievement in online learning with a greatly decreased number of parameters to learn when compared with existing distributed learning methods.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, Search; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms, Performance, Experimentation

Keywords

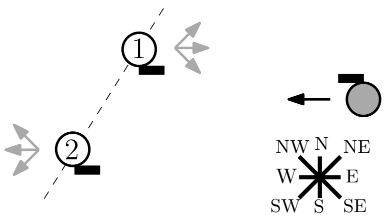
distributed; relational; multi-agent; reinforcement learning

1. INTRODUCTION

Reinforcement learning (RL) for the multi-agent setting is a challenging task. The state and action space of the learning problem increases exponentially with the number of agents. This gives rise to the problem of high model complexity that usually translates to a large number of parameters to be learned. In turn, online RL requires more time for exploration of the environment, jeopardizing overall goal achievement as less time is spent on exploiting learned knowledge.

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

(a)



(b)

Figure 1: (a) Example tactical RTS game. (b) Two white marines and their actions (gray arrows) that lead to *NotAligned* being true with respect to the oncoming gray enemy.

Relational representations [8, 10, 19, 6] have the potential to greatly reduce the number of learning parameters for cooperative multi-agent learning problems. They are also an intuitive way of encoding background knowledge to a problem. In particular, relational temporal difference (TD) learning [3] generalizes propositional state based features that are defined from relational predicates by combining them to give a single relational feature (RF). Such features can be used to model the value function of a policy with less parameters to learn.

Let us illustrate the generalization capability of RFs. Fig. 1a shows an example from the tactical real-time strategy (RTS) domain where the goal is to destroy the enemy team of marines. We use capital letters to denote first order predicate variables, and small letters to denote bound variables. Suppose we have a predicate that is true whenever two marines are not aligned relative to their nearest enemy as follows:

$$\begin{aligned} NotAligned(S_x, S_y, A_x, A_y) &:= SameNearestEnemy(S_x, S_y) \\ &\wedge\ DistanceIncreased(S_x, A_x, S_y, A_y) \end{aligned} \tag{1}$$

where S_x, S_y are the state variables of any agents x and y respectively, and A_x, A_y are their action variables, *SameNearestEnemy* indicates that the enemy nearest to the two marines are the same, while *DistanceIncreased* indicates that the difference of each marines' distance to their nearest enemy increases after taking actions A_x and A_y.

Fig. 1b shows the actions (indicated by the gray arrows) that will result in $NotAligned(s_1, s_2, a_1, a_2)$ being true for marines 1 and 2. This allows the system to learn to penalize actions that will result in marine 1 being further unaligned with marine 2. Such actions are undesirable as marines 1 and 2 will not be able to shoot at their nearest common enemy at the same time.

To model this knowledge of alignment, traditional RL systems may employ a linearly approximated action value function, $Q(\mathbf{s}, \mathbf{a}) \approx \sum_i w_i \rho_i(\mathbf{s}, \mathbf{a})$, where ρ_i is a propositional feature. For example, *NotAligned* propositions defined for all pairs of agents gives:

$$\begin{aligned}
&\rho_1(\mathbf{s}, \mathbf{a}) = \mathit{NotAligned}(s_1, s_2, a_1, a_2),\\
&\rho_2(\mathbf{s}, \mathbf{a}) = \mathit{NotAligned}(s_1, s_3, a_1, a_3),\\
&\rho_3(\mathbf{s}, \mathbf{a}) = \mathit{NotAligned}(s_2, s_3, a_2, a_3), \text{ etc} \cdots,
\end{aligned}$$

where $\mathbf{s} = \langle s_1, ..., s_N \rangle$ and $\mathbf{a} = \langle a_1, .., a_N \rangle$ denote the states and actions for N agents respectively. Hence, the system learns a weight for each pair of agents x, y.

In contrast, if we use a relational approach and make use of predicate *NotAligned* as an RF, ϱ', that is based on the count of valid bindings of agents to *NotAligned*, the RF is given by:

$$\varrho'(\mathbf{s}, \mathbf{a}) = \sum_{x \in [1,N]} \sum_{y \in [1,N]} \mathit{NotAligned}(s_x, s_y, a_x, a_y) \qquad (2)$$

We need only learn a single weight for ϱ'. Consequently, the experience from pairs of agents will contribute to the same weight and learning can be generalized to all pairs of agents x, y. This makes learning efficient, especially when the concept of alignment is similarly important between any pair of marines. Further, this form of RFs is flexible as it allows them to be combined with other kinds of features for function approximation. RFs also allow easy generalization of the learned parameters to closely related situations. For example, the parameter learned for ϱ' can be useful for situations involving teams of marines of varying sizes such as, four versus four, or six versus five marines.

To date, the use of relations are mostly limited to centralized RL where a designated controller is tasked to compute and store the learned parameters [10, 7]. Unfortunately, in a highly dynamic decentralized scenario, there is no such controller as the communication links among agents change over time. For example, in the RTS game, marines lose communication when destroyed or when out of range. Thus it is important that the learning system be distributed such that critical components do not reside in any one agent.

This paper addresses the challenge of providing relational generalization in a distributed environment. We propose a novel distributed approach to relational TD learning where there are dynamic communication links between agents. Our approach first obtains RFs from predicates that are *local* to each agent. Unlike the global centralized RFs, local RFs allow an agent based decomposition of the global relational value function. This scheme enables agents to generalize their interactions with other agents within themselves, and greatly decreases the number of parameters to be learned. Furthermore, there are no critical agents in the learning system. Second, message passing between neighboring agents is used to transfer current knowledge (learned weights) between agents' features that share relational semantics. This allows learning from interactions within a group of agents to be transferred to other groups whenever agents in separate groups come into contact with each other.

To the best of our knowledge, this is the first work that incorporates RFs involving multiple agents for distributed TD learning. We further describe how the proposed approach can be applied to value function based RL methods. Experiments show that the utilizing distributed relational generalizations improves the learning rates of existing distributed RL methods [13, 14] and reduces the number of parameters to learn. These results illustrate the utility of our approach, especially on domains where we expect agents to benefit from sharing learned knowledge with relational semantics.

2. PRELIMINARIES

We begin with the problem formulation. Given N number of agents, a decentralized Markov decision process (DEC-MDP) [4] is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ where

1. $\mathcal{S} = S_0 \times S_1 \times ... \times S_N$ is the joint state space consisting of state variable $s_0 \in S_0$ that is observable by all agents and variables $s_1 \in S_1, ..., s_N \in S_N$ that are local for each agent. A state is $\mathbf{s} = \langle s_0, s_1, ..., s_N \rangle \in \mathcal{S}$.
2. $\mathcal{A} = A_1 \times ... \times A_N$ is the joint action space with one variable for each agent. A joint action is $\mathbf{a} = \langle a_1, ..., a_N \rangle \in \mathcal{A}$.
3. $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the global transitional model, i.e., $\mathcal{P}(\mathbf{s}, \mathbf{a}, \mathbf{s}') = Pr(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ is the probability of reaching state $\mathbf{s}'$ by taking $\mathbf{a}$ in $\mathbf{s}$.
4. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ is the global reward function, giving the reward for taking action $\mathbf{a}$ in state $\mathbf{s}$ and reaching state $\mathbf{s}'$. $\mathcal{R}$ is decomposable into a sum of local rewards for each agent, i.e., $\mathcal{R}(\mathbf{s}, \mathbf{a}, \mathbf{s}') = \sum_{x=1}^{N} R_x(\mathbf{s}, \mathbf{a}, \mathbf{s}')$.

The communication structure defined by a state $\mathbf{s}$ can be represented as a coordination graph (CG) where agents are vertices and edges represent communication links between agents. Let $\Gamma(s_x)$ be the set of neighbors for an agent x identifiable through its state s_x. Each agent x may communicate with its neighbors $y \in \Gamma(s_x)$ and access their state and action variables. In general, a CG may consist of disjoint sub-graphs. Further, agents within the same sub-graph may synchronize their action selection by sending messages.

2.1 Distributed Model-Free RL

In model-free RL, the transition model $\mathcal{P}$ is neither learned nor required to find the policy (solution) to the DEC-MDP. Instead the global policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ is indirectly expressed as an action value function that encodes the expected return of taking an action in the current state while following π with discount rate γ , i.e., it is the Bellman equation

$$Q^\pi(\mathbf{s}, \mathbf{a}) = \sum_{\mathbf{s}' \in \mathcal{S}} \mathcal{P}(\mathbf{s}, \mathbf{a}, \mathbf{s}')[\mathcal{R}(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma Q^\pi(\mathbf{s}', \pi(\mathbf{s}'))]. \quad (3)$$

Then, policies can be computed using Q^π, e.g., for the greedy policy, $\pi(\mathbf{s}) = \operatorname{argmax}_{\mathbf{a}} Q^\pi(\mathbf{s}, \mathbf{a})$.

For distribution in a DEC-MDP, there are a variety of approaches to express the function Q^π in local parts [5, 17, 9, 11] with different assumptions about coordination. In particular, agent-based decomposition [13] additively decomposes Q^π into one local action value function per agent x,

$$Q^\pi(\mathbf{s}, \mathbf{a}) = \sum_{x=1}^{N} Q_x^\pi(\mathbf{s}_x, \mathbf{a}_x), \qquad (4)$$

where $\mathbf{s}_x$ and $\mathbf{a}_x$ are the respective projections of the global $\mathbf{s}$ and $\mathbf{a}$ on the variables that agent x may access in state $\mathbf{s}$. Suppose each agent's local function is a linear function approximation, $Q_x^\pi(\mathbf{s}_x, \mathbf{a}_x) \approx \sum_i w_{i,x} \phi_{i,x}(\mathbf{s}_x, \mathbf{a}_x)$. Then, each agent x performs on-policy TD updates for each weight $w_{i,x}$ locally with the step size parameter α,

$$w_{i,x} \leftarrow w_{i,x} + \alpha[r_x + \gamma Q^{\pi}_x(\mathbf{s}'_x, \mathbf{a}'_x) - Q^{\pi}_x(\mathbf{s}_x, \mathbf{a}_x)]\phi_{i,x}(\mathbf{s}_x, \mathbf{a}_x) \quad (5)$$

where $\mathbf{s}'_x$ and $\mathbf{a}'_x$ are the respective next state and action, and r_x is the observed decomposed reward for agent x. In general, TD updating makes use of incremental data between two consecutive time points to estimate the value function.

2.2 Coordination Guided RL

The work in [14] enabled background coordination knowledge to be used as a means to guide exploration via a system called distributed coordination guided RL (CGRL). Such coordination knowledge takes on the form of a set of coordination constraints (CCs). Each CC when *activated*, prevents the system from selecting certain actions in the state. Deciding when to activate constraints is part of the learning process, hence CCs are dynamic. This mechanism was realized through a distributed two-level RL system where the top level learned a policy to activate a subset of the CCs while the bottom level learned the actual actions to perform in the environment subject to the activated CCs.

The top level joint action space, $\mathcal{A}_0$, consists of one action variable for each CC with the domain $\{activate, deactivate\}$. The global greedy top and bottom policies are, $\pi^{\top}(\mathbf{s}) = \text{argmax}_{\mathbf{b}\in\mathcal{A}_0} W(\mathbf{s}, \mathbf{b})$ and $\pi^{\perp}(\mathbf{b}, \mathbf{s}) = \text{argmax}_{\mathbf{a}\in\mathcal{A}(\mathbf{b},\mathbf{s})} U(\mathbf{s}, \mathbf{a})$, where W and U are global top and bottom action functions respectively, $\mathbf{b} \in \mathcal{A}_0$ is a top joint action that selects a subset of CCs to be activated, and $\mathcal{A}(\mathbf{b}, \mathbf{s})$ is the original joint action space constrained by the top action $\mathbf{b}$ in state $\mathbf{s}$. Both action functions were distributed similarly like in Eq. (4) to give local linearly approximated functions, $W_x(\mathbf{s}_x, \mathbf{b}_x) \approx \sum_j w^W_{j,x}\phi^W_{j,x}(\mathbf{s}_x, \mathbf{b}_x)$ and $U_x(\mathbf{s}_x, \mathbf{a}_x) \approx \sum_i w^U_{i,x}\phi^U_{i,x}(\mathbf{s}_x, \mathbf{a}_x)$. Then, the weights $w^W_{j,x}$ and $w^U_{i,x}$ were estimated online using TD updating.

CCs are closely related to predicates used for RFs as they are negated propositions created from binding the predicates to specific agent variables. For example in Fig. 1b, we can specify the $\neg \textit{NotAligned}(s_1, s_2, a_1, a_2)$ proposition as a CC for agents 1 and 2. When this CC is activated, the bottom level action selection is subjected to the constraint that the agents may not take any of the actions specified by the gray arrows in Fig. 1b. If the top level of the system activates the *NotAligned* CCs frequently, exploration will be directed more often towards states where the agents are aligned.

Propositional features (PFs) for the agent based top level action functions W_x were also created by predicates. For example, a top level predicate may be defined to learn if it is better to deactivate the *NotAligned* constraint to allow a wounded agent flexibility to hide behind others as follows. Let $B_{x,y}$ be the action variable referring to the *NotAligned* CC for agent x and agent y. Then $\textit{Pred}(S_x, S_y, B_{x,y}) := [\textit{Wounded}(S_x) \vee \textit{Wounded}(S_y)] \wedge \textit{Deactivated}(B_{x,y})$, where *Deactivated* is true if the value of $B_{x,y}$ is *deactivated*. It is straightforward to see that in the case of a centralized controller, if *Pred* is used as an RF like in Eq. (2), as opposed to $N(N-1)/2$ weights for PFs, we can reduce the number of weights to one. This allows learning to activate *NotAligned* for a pair of agents to be generalized to any pairs of agents in the centralized case. Hence there is great potential to improve distributed CGRL if similar relational generalizations can be devised for the distributed case.

3. DISTRIBUTED RELATIONAL GENERALIZATIONS

In this section, we first present the centralized model-free relational TD learning. Then we describe how it is distributed using an approach internal to each agent and an external approach that requires passing messages.

We adapt the work in [3] for centralized TD learning using RFs for the action value function. We define agent arity as the number of unique agents from which the variables of a predicate come from. For example, the predicate $\textit{EgPred}(S_x, A_x, S_y, A_y)$ for any two agents x and y has an agent arity of $\eta(\textit{EgPred}) = 2$. Let ϱ_ρ represent an RF based on the predicate ρ. Further let the function $\textit{Perm}(N, n)$ return the set of n-permutations from the set of all N agents where the number of permutations is $|\textit{Perm}(N, n)| = {}_N\mathrm{P}_n = \frac{N!}{(N-n)!}$. For a predicate ρ, the construction of an RF for N agents in the centralized case is

$$\varrho_\rho(\mathbf{s}, \mathbf{a}) = \tau_\rho \cdot \sum_{i=1}^{{}_N\mathrm{P}_n} \rho(s_{i_1}, ..., s_{i_n}, a_{i_1}, ..., a_{i_n}) \quad (6)$$

where $n = \eta(\rho)$, τ_ρ is a scaling factor for predicate ρ, and $s_{i_1}, ..., s_{i_n}$, and $a_{i_1}, ..., a_{i_n}$ are projections on $\mathbf{s}$ and $\mathbf{a}$ respectively, ordered by some unique i-th n-permutation of the agents. Eq. (6) states that predicates may consist of the state variables and action variables of $n \leq N$ number of agents. When the $\tau_\rho = 1$, Eq. (6) describes the RF as the count of number of true propositions for every valid binding of variables to the predicate ρ.

Let F be the set of predicates. Then the global relational action value function, approximated as a linear function, is

$$Q(\mathbf{s}, \mathbf{a}) = \sum_{\rho\in F} w_\rho \varrho_\rho(\mathbf{s}, \mathbf{a}) \quad (7)$$

and the weights can be updated globally using

$$w_\rho \leftarrow w_\rho + \alpha[\mathbf{r} + \gamma Q(\mathbf{s}', \mathbf{a}') - Q(\mathbf{s}, \mathbf{a})]\varrho_\rho(\mathbf{s}, \mathbf{a}) \quad (8)$$

for a centralized system where $\mathbf{r}$ is the global reward.

Next, we describe how partial relational generalization is achieved for the individual agent's local action value functions for the distributed case. Without loss of generality, we take the presence of the action variable a_x to also indicate the presence of the accessible state variables $\mathbf{s}_x$.

3.1 Internal Relational Generalization

For distributed RL, each agent x carries the local Q_x function that are approximated using PFs. We observe that agents may internally generalize the weights between related PFs involving themselves and other agents. This is achieved by the partial binding of predicates to give RFs local to each agent. The result enables the sharing of learned parameters from an agent x's interactions with another agent y in a particular state with x's interaction with other agents $z \neq y$.

Consider the *NotAligned* predicate in Eq. (1). Suppose we have an RTS game with 4 agents and each agent has PFs created from predicates that involve themselves. Then agent 1 will have the pairwise PFs,

$$\{\textit{NotAligned}(a_1, a_2), \textit{NotAligned}(a_1, a_3), \textit{NotAligned}(a_1, a_4)\},$$

and agent 2 has

$$\{\textit{NotAligned}(a_1, a_2), \textit{NotAligned}(a_2, a_3), \textit{NotAligned}(a_2, a_4)\},$$

and likewise for the other agents. A partially bound predicate may be defined based on *NotAligned* by fixing the first agent variable such that $x = 1$, e.g., $\mathit{NotAligned}(a_1, A_y)$ where y is any other agent. Then, the internal *NotAligned* RF for marine x is

$$\varrho_{NotAligned,x}(\mathbf{a}_x) = \sum_{y \in [1,4]-\{x\}} \mathit{NotAligned}(a_x, a_y), \tag{9}$$

where *NotAligned* is assumed to be symmetrical, i.e., $\mathit{NotAligned}(a_x, a_y) = \mathit{NotAligned}(a_y, a_x)$.

This example indicates that agent 1 may generalize its knowledge of *NotAligned* from interaction with agent 2 to the other agents 3 and 4 as these interactions contribute towards updating the same feature weight. Furthermore, for N agents, we see that generalizing internally for pairwise predicates reduces the quadratic in N number of feature weights to a linear in N number of weights to be learned.

In general, for each predicate ρ, the partially bound, agent based local RF for agent x is defined as

$$\varrho_{\rho,x}(\mathbf{a}_x) = \frac{\tau_\rho}{n} \sum_{j=1}^{n} \sum_{i=1}^{{}^{N-1}\mathrm{P}_{n-1}} \rho(\dot{\mathbf{a}}_{x,j,n}) \tag{10}$$

where there are N agents, $n = \eta(\rho)$, $\dot{\mathbf{a}}_{x,j,n} = \langle a_{i_1}, ..., a_{i_{j-1}}, a_x, a_{i_{j+1}}, ..., a_{i_{n-1}} \rangle$ is a n-permutation of the values in the tuple $\mathbf{a}_x$, and τ_ρ is a scaling factor which we will assume to be 1 from here on. The double summations indicate that the variable a_x is inserted at the various positions j of the predicate ρ with respect to each $(n-1)$-permutation drawn from the other $N-1$ agents. That is, while the other $n-1$ agents' action variables may be drawn from the set of all other $N-1$ agents, agent x's participation in the RF is fixed. Note that the predicate is false for agent action values not found in $\mathbf{a}_x$, i.e., $\rho(\cdot)$ returns false (zero) for agents that are out of communication range.

With Eq. (10) and a set of predicates F, local action value functions can be specified for each agent x as a linear function approximation,

$$Q_x(\mathbf{a}_x) = \sum_{\rho \in F} w_{\rho,x} \varrho_{\rho,x}(\mathbf{a}_x) = \sum_{n=1}^{N} \sum_{\rho \in F_n} w_{\rho,x} \varrho_{\rho,x}(\mathbf{a}_x) \tag{11}$$

where $F_n \subseteq F$ is the set of predicates $\{\rho \in F \mid \eta(\rho) = n\}$ that is a partition of F such that $F = \bigcup_{n=1}^{N} F_n$. Eq. (11) essentially groups the sums based on the agent arity of the RFs, assuming that the maximum agent arity is N. Each $w_{\rho,x}$ may then be updated using Eq. (5). Next, we present the theoretical result that shows the agent based decomposition in Eq. (11) is an additive decomposition of the global relational action value function in Eq. (7).

THEOREM 1. *Given the set of predicates F and N agents, $Q(\mathbf{s}, \mathbf{a}) = \sum_{x=1}^{N} Q_x(\mathbf{s}_x, \mathbf{a}_x)$ if $w_\rho = w_{\rho,x}$ for all $\rho \in F, x \in [1, N]$, where $w_{\rho,x}$ is the local weight of the local RF $\varrho_{\rho,x}$ and w_ρ is the global weight for the global RF ϱ_ρ.*

PROOF. Assuming that the scaling factor for local and global value functions is $\tau = 1$, we have

$$\sum_{x=1}^{N} Q_x(\mathbf{a}_x) = \sum_{x=1}^{N} \sum_{n=1}^{N} \sum_{\rho \in F_n} w_{\rho,x} \varrho_{\rho,x}(\mathbf{a}_x) \tag{12}$$

$$= \sum_{n=1}^{N} \sum_{\rho \in F_n} \sum_{x=1}^{N} \frac{w_{\rho,x}}{n} \sum_{j=1}^{n} \sum_{i=1}^{{}^{N-1}\mathrm{P}_{n-1}} \rho(\dot{\mathbf{a}}_{x,j,n}) \tag{13}$$

$$= \sum_{n=1}^{N} \sum_{\rho \in F_n} w_\rho \sum_{x=1}^{N} \sum_{j=1}^{n} \sum_{i=1}^{{}^{N-1}\mathrm{P}_{n-1}} \frac{\rho(\dot{\mathbf{a}}_{x,j,n})}{n} \tag{14}$$

$$= \sum_{n=1}^{N} \sum_{\rho \in F_n} w_\rho \sum_{i=1}^{{}^{N}\mathrm{P}_{n}} \sum_{j=1}^{n} \frac{\rho(a_{i_1}, ..., a_{i_n})}{n} \tag{15}$$

$$= \sum_{n=1}^{N} \sum_{\rho \in F_n} w_\rho \varrho_\rho(a_{i_1}, ..., a_{i_n}) \tag{16}$$

$$= \sum_{\rho \in F} w_\rho \varrho_\rho(\mathbf{a}) = Q(\mathbf{a}) \tag{17}$$

□

Theorem 1 shows that the local relational agent based decomposition is able to represent the same function values as the ideal global case. This indicates that representational power is maintained when using the local value functions. Consequently, if the true global value function is well approximated by a linear combination of global RFs, we expect the same error bound to apply when using the local value functions with local RFs. Hence we have addressed the distributed requirement while retaining some relational generalization capability.

To represent relations, complex predicates can be constructed from logical operators on simpler predicates. Computing the valid bindings to give the grounded literals of these predicates, i.e., count of true bindings, may be performed by an inference engine or a customized implementation. Joint action selection with local Q_x functions approximated by local RFs can be solved using standard message passing methods [13, 18]. In our experiments, we use predicates with an agent arity of at most two. This results in efficient computation of the RFs.

3.2 External Relational Generalization

Internal generalization allows an agent to generalize learning between its individual interactions over various agents. While this introduces a form of shared experience and reduces the number of learning parameters in the system as a whole, there is still room for further generalization.

For example, consider the centralized *NotAligned* RF in Eq. (1). It allows all interactions between any two agents to share experience, i.e., whenever *NotAligned* for (a_1, a_2), (a_2, a_3), or (a_3, a_4) are true, they participate in updating the same weight. In contrast, Eq. (9) only allows agent 1's individual experience to be generalized within itself, i.e., whenever $\mathit{NotAligned}(a_1, a_2)$ or $\mathit{NotAligned}(a_1, a_4)$ are true they update the same weight in agent 1, but $\mathit{NotAligned}(a_2, a_3)$ and $\mathit{NotAligned}(a_3, a_4)$ do not play a part in that update when using the local RF in Eq. (10). Short of neighboring agents' influences during coordinated action selection, there is no other mechanism to pass on learned knowledge to other agents, as local updating shown in Eq. (5) does not involve the value functions (learned parameters) of other agents.

To further generalize the learned parameters of local RFs, we devise a message passing method to share the learned weights of the RFs based on the relations that they represent. From Theorem 1, the condition $w_{\rho,x} = w_\rho$ suggests that the goal of sharing is to tend towards equal weights for local RFs that are based on the same predicate.

In a synchronized message passing system, messages are passed for a number of iterations within each time step. At each iteration, agent x computes and sends the messages $\kappa_{x,y}(\rho)$ for each predicate, ρ, to its neighbor $y \in \Gamma(s_x)$,

$$\kappa_{x,y}(\rho) = c_{x,y} + w_{\rho,x} + \sum_{z \in \Gamma(s_x)-\{y\}} \kappa_{z,x}(\rho) \tag{18}$$

where $c_{x,y}$ is a normalizing value to ensure messages do not go to infinity in a graph with cycles [20]. The agent x sends the message $\kappa_{x,y}(\rho)$ to agent y that is an aggregation of the messages it received from its other neighbors, i.e., $\Gamma(s_x) - \{y\}$, from the previous iteration.

Once the desired iteration is achieved or the κ messages have converged, each agent x updates the weight $w_{\rho,x}$ for each local RF $\varrho_{\rho,x}$ with

$$w_{\rho,x} \leftarrow \beta \cdot [w_{\rho,x} + \sum_{y \in \Gamma(s_x)} \kappa_{y,x}(\rho)], \tag{19}$$

where β is used to normalize the messages.

In this message passing scheme, a fixed point exists in a CG without cycles. After applying Eq. (19) with $\beta = 1/N$, if the CG is connected, each local RF's weight will be the global average of all agents' weights for the RF. In the general case of a graph with cycles, the messages may not converge but the intermediary result from Eq. (19) at every iteration is usually useful in practice [13].

CGs are usually disjoint since agents rarely form a connected CG. Further, messages have to be sent for each RF's weight which may pose a problem when the number of RFs is large. To address these issues, we adapt an asynchronous message and update scheme as follows. In each time step, each agent x immediately sends the weight $w_{\rho,x}$ of the local RF $\varrho_{\rho,x}$ as a message to its neighbors. Upon receiving any message $w_{\rho,y}$ from a neighbor $y \in \Gamma(x)$, agent x immediately updates the weight of $\varrho_{\rho,x}$ using the convex combination,

$$w_{\rho,x} \leftarrow (1-\beta)w_{\rho,x} + \beta w_{\rho,y} = w_{\rho,x} + \beta[w_{\rho,y} - w_{\rho,x}] \tag{20}$$

where β controls the magnitude of the update. One added advantage of this scheme is that agents may send any subset of its weights as messages to its neighbors as permitted by its communication channel.

The update in Eq. (20) echoes that of the TD updates using a step size parameter α described in Eq. (5). The intent of both equations is to compute a form of averaging over streaming data. In the case of TD learning, the data comes in the form of one step transition and reward, where the goal is to estimate the expected return. In the case of external generalization, the data comes in the form of neighboring agents' weights and the goal is to estimate the mean of these weights.

External generalization messages can be sent in distributed TD learning algorithms right after the TD updating step. Our experiments apply internal generalization to the value functions of distributed RL and distributed CGRL described in Section 2. At each time step, after TD updates are performed, each agent sends their weights to their neighbors once. Received weights are combined using Eq. (20) according to their respective predicates ρ.

4. EXPERIMENTS

We carry out two sets of experiments to evaluate our proposed solution. The first set of experiments investigates the use of relational generalizations on distributed RL in two domains, simplified soccer and tactical RTS. Example predicates used are shown in Fig. 2 and 3 for soccer and RTS respectively. The second set of experiments demonstrate the utility of relational generalizations with distributed CGRL in the tactical RTS domain.

We implemented 3 RL methods: **(a)** *independent* refers to independent learners [5], **(b)** *coordinated* refers to distributed RL (Section 2.1), and **(c)** *DistCGRL* refers to distributed CGRL (Section 2.2). Relational generalizations are applied to both coordinated and DistCGRL learners with various values for the external generalization parameter, $\beta \geq 0$. $\beta = 0$ indicates that only internal generalization is used. All agents use ϵ-greedy policies for on-policy TD updating. Such policies select the maximum action with probability $1-\epsilon$ and a random action with probability ϵ.

$$\begin{aligned}
\textit{FlankOffensive}(A_x, A_y) &:= \textit{Flank}(A_x) \wedge \textit{Flank}(A_y) \\
&\wedge \textit{Forward}(A_x) \wedge \textit{Forward}(A_y) \\
\textit{Defensive}(A_x, A_y) &:= \textit{FrontQuarter}(A_x) \wedge \textit{FrontQuarter}(A_y) \\
&\wedge \textit{MidField}(moveTo(A_x)) \wedge \textit{MidField}(moveTo(A_y)) \\
\textit{Collide}(A_x, A_y) &:= moveTo(A_x) = moveTo(A_y) \\
\textit{GoodPass}(A_x, A_y) &:= \textit{IsPass}(A_x, A_y) \wedge \neg\textit{MoveWithinEnemyIntercept}(A_x)
\end{aligned}$$

Figure 2: Example soccer predicates

$$\begin{aligned}
&\textit{PairAttack}(A_x, A_y) := \textit{TargetInRange}(A_x) \wedge \textit{TargetInRange}(A_y) \\
&\qquad \wedge target(A_x) = target(A_y) \\
&\textit{PairCloser}(A_x, A_y) := \textit{Closer}(A_x, e) \wedge \textit{Closer}(A_y, e) \\
&\qquad \wedge \textit{SameNearestEnemy}(A_x, A_y) \wedge nearestEnemy(A_x) = e \\
&\textit{OneOfTwoAttack}(A_x, A_y) := \neg\textit{PairAttack}(A_x, A_y) \\
&\qquad \wedge [\textit{TargetInRange}(A_x) \vee \textit{TargetInRange}(A_y)] \\
&\textit{WeakerMoveCloser}(A_x, A_y) := \textit{SameNearestEnemy}(A_x, A_y) \\
&\qquad \wedge \textit{Closer}(weaker(A_x, A_y), nearestEnemy(A_x))
\end{aligned}$$

Figure 3: Example RTS predicates

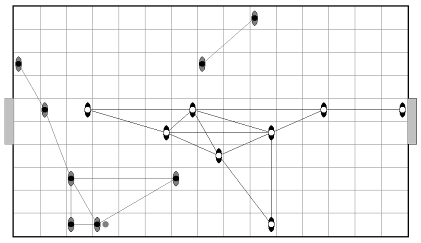


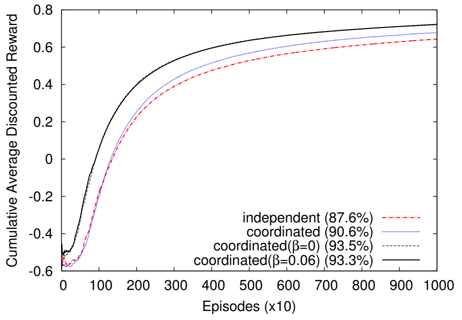
Figure 4: Example state in simplified soccer

RL Method	# Weights	% Weights
independent	1888	60.5
coordinated	3120	100.0
coordinated($\beta \geq 0$)	2064	66.2

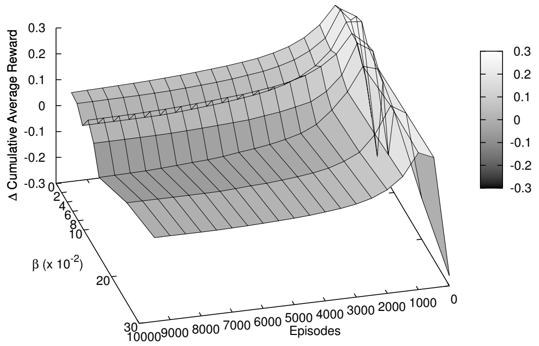
Table 1: Weights to learn for soccer

Fig. 4 shows an example state of the simplified soccer domain. A 15×10 grid is used with two teams of 8 players each, gray and black. The objective is to score the first goal. Players may move in 4 directions, and pass or shoot when in possession of the ball. Communication is allowed only at a Manhattan distance of 5. The lines in Fig. 4 describe the current CG based on the state. We pit the RL players against a scripted opponent that defends their goal and counterattacks once the ball is intercepted. The total number of weights to learn for function approximation for each type of learner is given in Table 1. The learners use $\gamma = 0.99$ with decaying step size, $\alpha = \{10^{-2}, 10^{-4}, 0.998\}$, and exploration, $\epsilon = \{1, 10^{-2}, 0.998\}$, parameters written as $param = \{\text{initial, final, decay rate}\}$. The player that scores a goal receives a reward of 1 while a reward of -1 is evenly divided among soccer players nearest to their home goal when the opponent scores.

Fig. 5a shows the results for online learning using cumulative average reward. Each plot is an average of 10 learning runs. We observe that internal relational generalization



(a) Best relational TD results, final win rate in brackets



(b) Change in reward over coordinated learner for various β

Figure 5: Soccer experiment results.

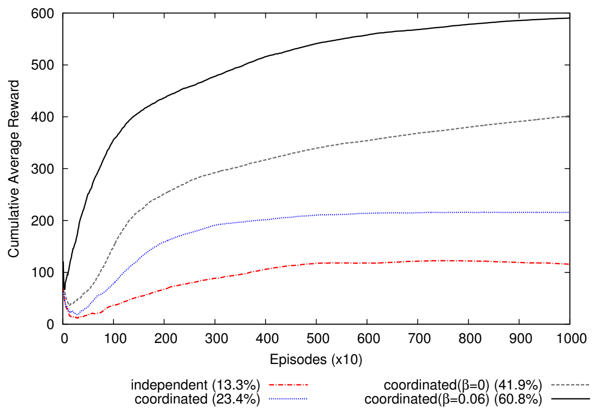
($\beta = 0$) is beneficial to soccer over the coordinated learner, while external generalization ($\beta = 0.06$) does not yield any benefit as its curve overlaps with $\beta = 0$. In Fig. 5b we plot the change in reward, i.e., the reward for the relational learners subtract the coordinated learner for each episode. We observe that both forms of relational generalization give rise to high benefit at the start. But, the benefit is lost over time for most settings of $\beta > 0$. This is expected as generalization allows fast initial learning, yet it is role specialization that gives a soccer team an edge over time. Nevertheless better performance is achieved using only 66.2% of the quantity of learning parameters of the coordinated player (see Table 1).

For the tactical RTS domain, we use a 240×240 point based world as shown in Fig. 1a. There are 10 RL marines, each may move in 8 directions and shoot at any enemy within range, thus the joint action space is massive. Marines may only communicate within a range of 30 points. We pit the RL marines against scripted marines that move towards their nearest enemy and start shooting. Rewards are -1 per time step and 1000 for winning that are equally divided among *surviving* marines. For RTS, learners used constant $\alpha = 10^{-4}$ and $\epsilon = \langle 0.5, 10^{-2}, 0.998 \rangle$. No discounting is used, i.e., $\gamma = 1$. The number of weights and RL parameters are shown in Table 2. Note that the use of internal generalization reduces the weights for the coordinated($\beta \geq 0$) learners to 26.9% of the coordinated learner. This is because most predicates are multi-agent, involving pairs of agents.

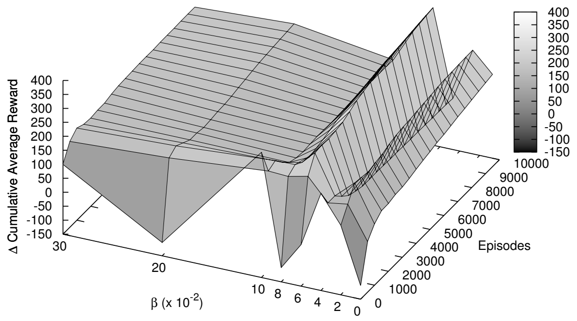
Fig. 6 shows the results for the coordinated learners versus 10 enemy marines. From Fig. 6a, it is clear that relational generalization is highly desirable in the tactical RTS domain where formation among marines is of paramount importance. We see improvement from internal generalization in the coordinated ($\beta = 0$) learner and further improvement when external generalization messages are exchanged. Fig. 6b shows the change in reward over the coordinated learner without RFs. We observe that distributed relational TD learning is beneficial for most values of β.

RL Method	# Weights Bottom	Top	Total	%
independent	350	0	350	17.8
coordinated	1970	0	1970	100.0
coordinated($\beta \geq 0$)	530	0	530	26.9
CentCGRL	53	45	98	2.1
DistCGRL	1970	2690	4660	100.0
DistCGRL($\beta \geq 0$)	530	450	980	21.0

Table 2: Weights to learn for RTS. Bottom indicates the weights for flat RL, e.g, the Q function, or the U function in CGRL while Top indicates the weights for top level function W in CGRL.



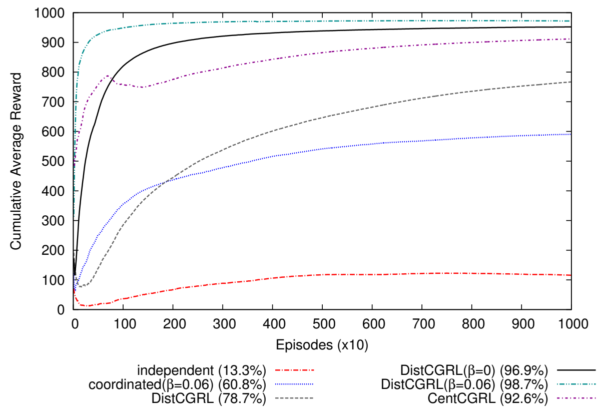
(a) Best relational TD results, final win rate in brackets



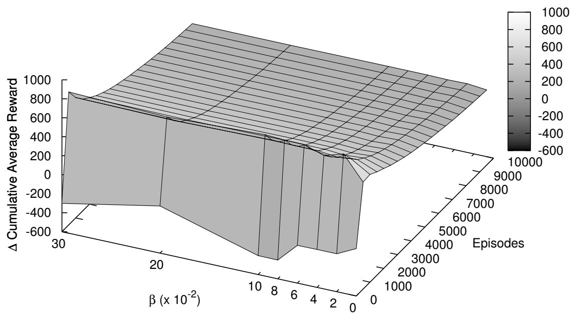
(b) Change in reward over coordinated learner for various β

Figure 6: RTS experiment against 10 enemy marines using coordinated learners for 10K episodes.

For the second set of experiments on distributed relational TD learning applied to DistCGRL, we use the same tactical RTS setup of 10 RL marines versus 10 scripted marines. Fig. 7 shows the results. We compared our results with centralized CGRL (CentCGRL) that made use of RFs [15]. From Fig. 7a, we observe that DistCGRL without RFs is superior to coordinated ($\beta = 0.06$), but its performance is nowhere near that of CentCGRL. However, once relational TD is introduced, DistCGRL outperforms CentCGRL and almost achieves a perfect win rate of 98.7% when $\beta = 0.06$, achieved with only 21% of the number weights in DistCGRL without RFs. The largest reduction comes from the top level function (see Table 2). The improved performance

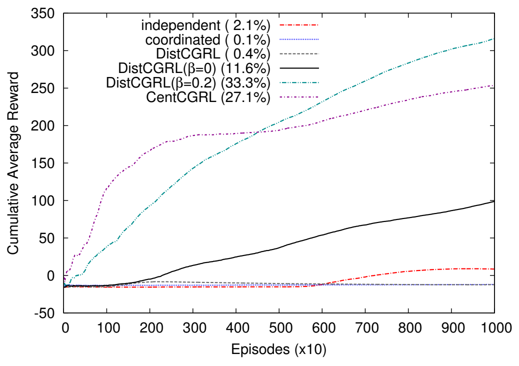


(a) Best relational TD results, final win rate in brackets

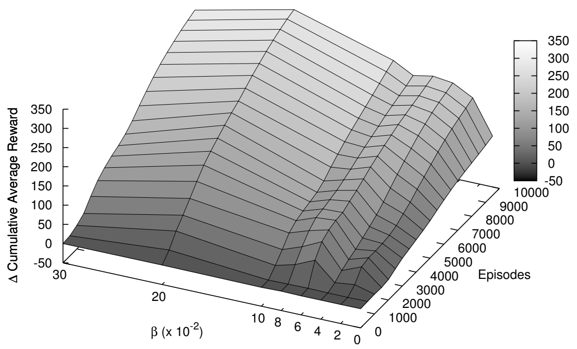


(b) Change in reward over DistCGRL learner for various β

Figure 7: RTS experiment against 10 enemy marines using DistCGRL learners for 10K episodes.



(a) Best relational TD results, final win rate in brackets



(b) Change in reward over DistCGRL learner for various β

Figure 8: RTS experiment against 13 enemy marines using DistCGRL learners for 10K episodes.

over CentCGRL may be due to agent decomposed rewards that provide fine-grained information for DistCGRL in contrast to the global reward scheme in CentCGRL. Generally, external generalization further improves learning for most values of β as shown in Fig. 7b.

To further demonstrate the benefits of our relational TD concepts, we pit the RL marines against a harder scenario with 13 enemy marines. The results are shown in Fig. 8. In this scenario, independent, coordinated, and DistCGRL methods without RFs are unable to learn to overcome their opponent within 10K episodes as shown in Fig. 8a. However, relational generalization eventually enables DistCGRL to learn better policies than CentCGRL. This illustrates that rapid generalization is crucial for a fast-paced domain like tactical RTS whereby marines are destroyed easily resulting in few samples of winning episodes to explore. In Fig. 8b, we see that external generalization improves learning throughout the episodes with the best improvement at $\beta = 0.2$.

5. DISCUSSION & RELATED WORK

Our experiments employed a coarse-grained approach towards the specification of local RFs for internal generalization by applying it to all predicates. We expect results to improve further especially for domains that require specialized roles, e.g., soccer, if the user carefully defines a subset of predicates to apply internal generalization. Furthermore, the results in Fig. 5b suggest that for domains like soccer, we should perform generalization early but decrease it over time to allow agents to develop specialized value functions.

For domains with homogeneous agents, such as tactical RTS, relational generalization allows experience to be shared rapidly. Results show better goal achievement while having less parameters (weights) to learn and store. In some cases the performance improvement is dramatic as we have seen in Fig. 7 using only 21% of the number of weights of a propositional DistCGRL approach.

Distributed relational TD learning may also be preferred for a centralized application when it is feasible to provide agent decomposed rewards from the MDP. From this perspective, the β parameter in external generalization gives the user some control over the amount of relational generalization for each agent's learning, i.e., agents may retain some form of individuality. Results in RTS have shown that it is possible to outperform even centralized approaches with RFs when using certain settings for β. Last, almost all settings of β demonstrate improvement over learning without external generalization, hence the user need not fine-tune the value of β if the current goal achievement is acceptable.

Relational generalization for RL arose with the need to compactly represent state and action spaces in combinatorial domains such as the blocks (un)stacking domain [8, 12]. We have used the general linear function approximation approach to model value functions. This is more akin to the work in [3] as opposed to relational regression tree family of methods used in [8, 7, 6]. The benefits of our approach include the fast update time that grows linearly with the number of weights, and the ability to include non-predicate based features. In the event that the user specifies poor predicates, the effect is similar to having poor features in function approximation that do not discriminate well between state action pairs. The work on multi-agent relational RL in [16] investigated learning to communicate with experts when communication between agents is costly, expert agents exists, and tasks are solved individually. In contrast, our approach begins with no known expert, agents solve their task jointly and may interfere with each other.

Our proposed method is different from other multi-agent RL works that incorporate parts of the local value functions of other agents within each agent as the relational semantics determines the experience to be shared among agents. In [17], agents' updates incorporate tabular Q-values from other agents' local value functions based on the current state and action values. However, in our approach agents may share learned weights for RFs that are not limited to the current state and action. [1, 2] showed that averaging tabular action value functions between agents can improve learning performance when some agents are experts . However, this is prohibitive for large value functions and heterogeneous agents that do not share the same state-action pairs.

The relational predicates are currently provided by the user. No distinction was made between the base level literals, that encode the basic relational representation for states and actions, from higher level predicates such as *NotAligned*. Both are regarded as similar background knowledge since we do not assume a readily encoded domain is available. Works in centralized single agent relational TD have discussed possible solutions for automated construction of higher level predicates from lower level predicates [12, 3]. Adapting these ideas to the communication requirements of the distributed environment will reduce dependence on the human expert.

6. CONCLUSION

To the best of our knowledge, this is the first work to bring the generalization capabilities of relational TD learning to the distributed case with dynamic communication. This was achieved using two parts, internal and external relational generalization. Internal generalization creates local RFs from partially bound predicates, enabling individual agents to generalize over their interactions with various agents and greatly reduces the number of parameters to learn in the system as a whole. External generalization involves message passing of learned parameters that share relational semantics, allowing different groups of agents to share their experience with others. Experiment results on two domains show that the proposed methods leads to better online learning through rapid generalization of experience. Exceptional results are achieved when the domain involves homogeneous agents like the tactical RTS domain. Relational generalizations were applied to various value function based RL methods to illustrate the general applicability of the proposed concepts. Furthermore, the competitive results with centralized approaches shows that the challenge of distribution can be resolved while retaining much of the benefits of relational TD learning.

Possible directions for future work include: **(a)** adaptively adjusting the amount of relational generalization over time for domains where specialized roles are important; **(b)** automatically creating new predicates in a distributed environment; **(c)** extending this work to domains with varying communication cost or where communication has to be explicit from agents' actions.

7. ACKNOWLEDGMENTS

This work was supported by A*STAR Exploit Flagship Grant ETPL/10-FS0001-NUS0.

8. REFERENCES

[1] M. Ahmadabadi and M. Asadpour. Expertness based cooperative Q-learning. *IEEE Trans. Syst., Man, Cybern., Syst., Part B*, 32(1):66 –76, 2002.

[2] B. Araabi, S. Mastoureshgh, and M. Ahmadabadi. A study on expertise of agents and its effects on cooperative Q-learning. *IEEE Trans. Syst., Man, Cybern., Syst., Part B*, 37(2):398 –409, 2007.

[3] N. Asgharbeygi, D. J. Stracuzzi, and P. Langley. Relational temporal difference learning. In *ICML*, pages 49–56, 2006.

[4] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.*, 27:819–840, 2002.

[5] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI/IAAI*, pages 746–752, 1998.

[6] T. Croonenborghs, J. Ramon, H. Blockeel, and M. Bruynooghe. Online learning and exploiting relational models in reinforcement learning. In *IJCAI*, pages 726–731, 2007.

[7] T. Croonenborghs, K. Tuyls, J. Ramon, and M. Bruynooghe. Multi-agent relational reinforcement learning. In *LAMAS*, pages 192–206, 2005.

[8] S. Džeroski, L. De Raedt, and K. Driessens. Relational reinforcement learning. *Mach. Learn.*, 43(1/2):7–52, 2001.

[9] E. Ferreira and P. Khosla. Multi agent collaboration using distributed value functions. In *IEEE Int. Veh. Sym*, pages 404 –409, 2000.

[10] C. Guestrin, D. Koller, C. Gearhart, and N. Kanodia. Generalizing plans to new environments in relational MDPs. In *IJCAI*, pages 1003–1010, 2003.

[11] C. Guestrin, M. G. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *ICML*, pages 227–234, 2002.

[12] K. Kersting and L. De Raedt. Logical Markov decision programs and the convergence of logical TD(λ). In *ILP*, volume 3194 of *LNCS*, pages 103–116. Springer, 2004.

[13] J. R. Kok and N. Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *J. Mach. Learn. Res.*, 7:1789–1828, 2006.

[14] Q. P. Lau, M. L. Lee, and W. Hsu. Distributed coordination guidance in multi-agent reinforcement learning. In *ICTAI*, pages 456–463. IEEE Computer Society, 2011.

[15] Q. P. Lau, M. L. Lee, and W. Hsu. Coordination guided reinforcement learning. In *AAMAS*, volume 1, pages 215–222. IFAAMAS, 2012.

[16] M. Ponsen, T. Croonenborghs, K. Tuyls, J. Ramon, K. Driessens, J. van den Herik, and E. Postma. Learning with whom to communicate using relational reinforcement learning. In *ICIS*, volume 281 of *SCI*, pages 45–63. 2010.

[17] J. G. Schneider, W.-K. Wong, A. W. Moore, and M. A. Riedmiller. Distributed value functions. In *ICML*, pages 371–378, 1999.

[18] R. Stranders, F. M. D. Fave, A. Rogers, and N. R. Jennings. A decentralised coordination algorithm for mobile sensors. In *AAAI*, pages 874–880, 2010.

[19] P. Tadepalli, R. Givan, and K. Driessens. Relational reinforcement learning: An overview. In *ICML Workshop on Relational RL*, pages 1–9, 2004.

[20] M. Wainwright, T. Jaakkola, and A. Willsky. Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. *Statistics and Computing*, 14(2):143–166, 2004.