	$o_1^{i:k}$	$o_2^{i:k}$
$o_1^{j:l}$	u, v	x, y
$o_2^{j:l}$	y, x	v, u

(a)

	$o_1^{i:k}$	$o_2^{i:k}$
$o_?^{j:l}$	?, ?	?, ?
$o_?^{j:l}$	?, ?	?, ?

(b)

Figure 1: Payoff matrices for 2×2 games

Second, given a pair of tasks (T_k, T_l) these may be *incompatible* if performed by a single agent, given joint options available. For instance, the tasks of meeting someone and working on a paper are incompatible if performed by the same agent during a specific time period. Therefore incompatible tasks require a kind of compatibility between options, which is assumed to be known by each agent for its *own* tasks. We denote the *compatibility* relation between options for performing tasks (T_k, T_l) by $\diamond_l^k$ defined in $\mathcal{O} \times \mathcal{O}$. There is no relation assumed between the sets $\sim$ and $\diamond$.

Given these operational constraints, agents must select strategies (or policies) among the available options, which are (a) consistent for performing tasks jointly with others, and (b) are compatible for performing own tasks.

A social context for an agent A_i ($SocialContext(A_i)$) is the set of tasks played by the agents in its neighborhood: $SocialContext(A_i) = \{T_k | T_k \in \mathcal{T} \wedge \exists A_j \in N(A_i)\ s.t.\ j:k\}$.

Thus, $SocialContext(A_i)$ includes agents' own tasks. Also, all tuples of tasks in $Tasks(A_i) \times (SocialContext(A_i) - Tasks(A_i))$ should be performed jointly by A_i and one of its peers in $N(A_i)$. Considering two acquaintances A_i and $A_j \in N(i) - \{A_i\}$, and two tasks T_k and T_l that must be performed jointly by these agents (i.e. it holds that $[A_i{:}T_k \bowtie A_j{:}T_l]$), agents must select among the sets of available options $O_{i:k}$ and $O_{j:l}$ respectively, so as to increase their expected payoff with respect to their task-specific preferences on options and operational constraints.

Given, for instance, two options per agent and task, agents are assumed to play a game of the form shown in figure 1(a): All entries in this matrix are different than zero, x, y are positive integers, and u, v are negative integers. As it can be noticed, this can be a coordination game, if $u = v$, $x = y$, with two Nash equilibria, namely the joint options proving payoffs x and y. However, this is not necessarily a symmetric game, given that the payoff received depends on the task performed by each agent. It must be noticed that any agent in the society plays such a game for any combination of tasks to be performed jointly with peers (i.e. for every task in its social context).

Given that the consistency relation among options available to agents for performing their tasks is not known to any in the society, and that agents do not know about the payoffs of other agents when choosing specific strategies to perform their tasks, agents need to learn about the structure of the game to be played, and they have to coordinate with others, as well. The information that an agent (e.g. the row playing agent) has about a 2×2 game is as shown in figure1(b). Question marks indicate the missing information: For instance, the agent does not know neither how options can be combined, nor the payoffs from these combinations.

For conventions to emerge in the society, any pair of connected agents (anywhere in the society) jointly performing tasks T_k, T_l must reach the same decisions for their strategies to perform their tasks.

This problem specification emphasises on the following aspects of the problem:

- Agents need to coordinate their strategies (i.e. chosen options) to perform own and joint tasks w.r.t. their preferences and operational constraints.
- Agents need to explore and discover how different combinations of options affect the performance of joint tasks w.r.t. operational constraints, given that consistencies among options are not known: This is essential for them to coordinate effectively and finally reach conventions.
- Agents must coordinate and reach agreements on the strategies to be followed for performing their joint tasks, w.r.t. their preferences, operational constraints and assumptions on options' consistency.
- Agents' preferences on the options available may vary depending on the task performed and are kept private.
- Each agent must learn to perform multiple (even incompatible), interrelated tasks in its social context.

3. REINFORCEMENT LEARNING METHODS FOR COMPUTING CONVENTIONS

To describe the proposed methods for the computation of conventions, we distinguish between two, actually highly intertwined, computation phases: (a) The computation of consistencies between options for joint tasks in the social context of each agent, and the computation of strategies for performing tasks w.r.t. own preferences, operational constraints and strategies of peers; and (b) the computation of contextual agreements concerning agents' strategies to perform joint tasks.

Given an agent A_i performing the task T_k, and any task $T_l \in SocialContext(A_i)$ assigned to an agent A_j, it holds that $[A_i{:}T_k \bowtie A_j{:}T_l]$. Agents need to coordinate towards performing these tasks jointly. To do that, agents need to discover the consistencies between the options in $O_{i:k}$ and $O_{j:l}$.

The agent A_i (e.g. based on the information disclosed to it) may assume a subjective consistency relation between available options, denoted by $\sim_i$. Given that relation, A_i has to make a specific decision for the option to be chosen to perform its own tasks, w.r.t (known) compatibility restrictions, contributing also to the coordinated performance of tasks. This choice of A_i, among the available options in $O_{i:k}$, is the *strategy* of the agent to perform the task T_k. This strategy is denoted by *str(i:k*).

As already said, each task assigned to an agent A_i is an own task which must be performed jointly with peers. Therefore, given a pair of incompatible tasks T_k, T_l assigned to A_i, the following validity constraint holds:

$$str(A_i{:}T_k) \diamond_l^k str(A_i{:}T_l),$$

In other words, strategies chosen for incompatible tasks must be compatible. Of course, considering that these tasks are performed jointly with other tasks in the social context of agents, they must also respect consistency constraints. Given this validity constraint among strategies, as well as the compatibility relation $\diamond$ among own tasks, the utility *U(i:k,o)* of choosing the option $o \in O_{i:k}$ in case it holds that $A_i : T_k$ is

$$U(i{:}k, o^{i:k}) = \gamma_{i:k}(o) + f(i{:}k, o),$$

where $\gamma_{i:k}(o)$ is the preference of agent A_i for the option o towards performing T_k, and

$$f(i{:}k, o) = Profit * SatisfiedConstraints(i{:}k, o) + Penalty * ViolatedConstraints(i{:}k, o)$$

$Profit$ is a positive number representing the reward of any satisfied operational constraint in the social context of agent A_i and $Penalty$ is a negative number that represents the cost of violating a constraint. $SatisfiedConstraints(i{:}k, o)$ (resp. $ViolatedConstraints(i{:}k, o)$) is the number of satisfied (resp. violated) compatibility constraints for the tasks performed by the agent A_i.

3.1 The MDP framework.

Using the model of collaborative multiagent MDP framework [14], [10] we assume:

-The **society** of agents $S = (\mathcal{T}, \mathcal{A}, \mathcal{E})$.
-A **time step** $t = 0, 1, 2, 3, ...$
-A set of **state variables** per agent-task $A_i{:}T_k$ performed jointly with task T_l in $SocialContext(A_i)$ at time t, denoted by $s^t_{(i:k \bowtie \cdot:l)}$[2]. Each state variable corresponds to the combination of tasks and ranges to the sets of subjective consistent options for tasks T_k, T_l assumed by A_i, i.e. $s^t_{(i:k \bowtie \cdot:l)} \in 2^{O_{\cdot:k} \times O_{\cdot:l}}$, s.t. $s^t_{(i:k \bowtie \cdot:l)} \subseteq \sim_i$. The **local state** s^t_i of agent A_i at time t is the tuple of the state variables for all tasks assigned to A_i in combination with any task in its social context. A **global state** s^t at time t is the tuple of all agents' local states.
-A **strategy** per agent-task $A_i{:}T_k$ at time t is denoted by $str^t_{i:k}$. The **local strategy** for agent A_i at time t, denoted by str^t_i is a tuple of strategies, each for any task that A_i performs. The **joint strategy of a subset of agents** A of $\mathcal{A}$ accomplishing their tasks (for instance of $N(A_i)$) at time t, is a tuple of local strategies, denoted by str^t_A (e.g. $str^t_{N(A_i)}$). The set of all joint strategies for $A \subset \mathcal{A}$ is denoted $Strategy_A$. The **joint strategy** for all agents $\mathcal{A}$ at time t is denoted str^t.
-The **state transition function** gives the transition to the joint state s^{t+1} based on the joint strategy str^t taken in joint state s^t. Formally $Tr : State \times Strategy \rightarrow State$. It must be noticed that although this transition function may be deterministic in settings with perfect knowledge about society dynamics, the state transition per agent is stochastic, given that no agent has a global view of the society (and as it will be discussed, under limited monitoring abilities it does not have a complete view of its neighbourhood), and thus no agent can predict how the joint state can be affected in the next time step. Thus, for agent A_i this transition function is actually $Tr : State_i \times Strategy_{\{A_i\}} \times State_i \rightarrow [0, 1]$, denoting the transition probability $p(s^{t+1}_i | s^t_i, str^t_i)$.
-A **joint reward** denoted by $Rwd_{[i:k \bowtie \cdot:l]}$, specifies the reward received by A_i performing the task T_k jointly with agents performing the task T_l in $SocialContext(A_i)$. The **task-specific reward** for $A_i{:}T_k$, denoted by $Rwd_{i:k}$, specifies the individual reward provided to agent A_i performing the task T_k, based on the joint strategies of agents in $N(A_i)$. This is the sum of $Rwd_{[i:k \bowtie \cdot:l]}$, for any $T_l \in SocialContext A_i$. The **local reward** of agent A_i, denoted Rwd_i, is the sum of its rewards for all the tasks that it performs.

It must be noticed that states represent agents' assumptions about options' consistency, while agents' strategies concern the specific options chosen for performing tasks w.r.t. these assumptions and the compatibility relation.

Given that agents do not have prior knowledge about the consistency relation among options, they do not know about the joint effects of strategies chosen. This information has to be learned based on the rewards received (including feedback from others).

The joint reward $Rwd_{[i:k \bowtie \cdot:l]}$ depends on the utility of agents' options while accomplishing specific tasks, as defined above, on feedback received from peers concerning agents' strategy, and on the payoff received after performing the chosen strategy as part of a joint strategy. Formally:

$$\begin{aligned} Rwd_{[i:k \bowtie \cdot:l]}(s^t, str^t_{i:k}) = a * U(i{:}k, str^t_{i:k}) \\ + b * Feedback(s^t, str^t_{i:k}) \\ + Payoff(str^t_{i:k}). \end{aligned} \quad (1)$$

$$\begin{aligned} Feedback(s^t, str^t_{i:k}) = Profit * Feedback^+(s^t, str^t_{i:k})) \\ + Penalty * Feedback^-(s^t, str^t_{i:k})). \end{aligned} \quad (2)$$

where $s^t = s^t_{[i:k \bowtie \cdot:l]}$, i.e. the state variable for $[i{:}k \bowtie \cdot{:}l]$ and $str^t_{i:k}$ is the option deliberatively selected as strategy for $i{:}k$ at time t. $Feedback^+(s^t_i, str^t))$, $Feedback^-(s^t_i, str^t))$ are the numbers of positive and negative feedbacks received, respectively, from peers given the strategy selected and the assumptions about options' consistency formed by A_i, i.e. $\sim_i$. The way feedback is received and counted will be presented in detail in subsequent paragraphs. $Profit$ and $Penalty$ are the numbers specifying the profit and cost for each positive and negative feedback, respectively (being equal to the corresponding utility parameters). The parameters a and b have been used for balancing between own utility and feedback received by others: As previous works have shown [22], although the role of both is crucial for agents to reach agreements, the method is tolerant to different values of these parameters. Here we consider that $\frac{a}{b} = \frac{1}{10}$. Finally, $Payoff(str^t_{i:k})$ is the payoff received by the agent A_i when it performs T_k according to the strategy chosen at time t.

Thus, the reward received by any agent depends on (a) the operational constraints w.r.t the assumptions made on consistency constraints, and (b) the agreement between peers on the means to be used for performing tasks jointly. As far as the operational constraints are concerned, compatibility between own options for performing tasks affect the utility of the agent, while consistency of options for joint tasks is provided via the payoff received. Thus, it is expected that the payoffs received should steer agents towards deciding consistent strategies. Feedback received from peers show whether agents have reached agreement on the consistency of options to be used. Since agents do not know the relation $\sim$, feedback supports them to check whether their peers agree on their subjective assumptions $\sim_i$. This (in conjunction to payoffs received) allows more knowledgeable / informative agents to help on discovering the relation $\sim$.

A **(local) policy** of an agent A_i in its social context is a function $\pi_i : State_i \rightarrow Strategy_{\{A_i\}}$ that returns local strategies for any given local state, for all the tasks assigned to A_i. The objective for any agent in the society is to find an optimal policy π^* that maximizes the expected discounted future return $V^*_i(s) = max_{\pi_i} E[\sum_{t=0}^{\infty} \delta^t Rwd_i(\pi_i(s^t_i), s^t_i) | \pi_i)]$

[2]The notation $(\cdot{:}l)$ means "any agent performing the task T_l". When it appears in $(i{:}k \bowtie \cdot{:}l)$ means "any agent in $N(A_i)$ performing the task T_l".

for each state s_i, while performing its tasks. $\delta \in [0,1]$ is the discount factor.

This model assumes the Markov property, assuming also that rewards and transition probabilities are independent of time. Thus, the state next to state s is denoted by s' and it is independent of time.

3.2 Computing contextual agreements

The subjective assumptions on the consistency of available options (i.e. their state) for any tasks agents jointly perform may not agree. More importantly, the strategies chosen for the joint performance of tasks may be inconsistent to those chosen by peers. Towards reaching agreements on the consistency between available options (which is important for agents to select valid joint strategies), agents consider the feedback received from their peers. From now on, we use the term *decision* to indicate the combination of agent state and strategy chosen: Thus when agents revise their decisions they may revise their subjective assumptions on the consistency between options, or their strategies for performing tasks, or both.

According to this communication-based learning approach, given an agent A_i performing the task T_k, and a task $T_l \in SocialContext(A_i)$, i.e. $[A_i{:}T_k \bowtie {\cdot}{:}T_l]$, to get feedback on decisions about the performance of T_k, the agent A_i propagates its decision to its peers performing T_l. Actually, it forwards its strategy for performing T_k, i.e. $o_{i:k} = str^t_{i:k}$ and the[3] assumed consistent option for others to perform T_l, i.e. $o_{\cdot:l}$ s.t. $\langle o_{i:k}, o_{j:l}\rangle) \in \sim_i$. It must be noticed that A_i forwards its decision to all agents in $N(A_i)$ that have been assigned T_l. This happens for any such pair of tasks.

The decision propagated to a specific $A_j \in N(A_i)$ at any t is of the form $(A_i{:}T_k,\ A_j{:}T_l,\ \langle o_{i:k}, o_{j:l}\rangle)$, indicating the peers involved, the strategy chosen, and the subjective view of the sender for the consistent option of the recipient A_j to perform T_l.

Agents propagate their decisions to their acquaintances in the network iteratively and in a cooperative manner, aiming to exploit the transitive closure of correspondences in cyclic paths. This is similar to the technique reported in [22]. Agents propagate what we call $c-histories$, which are ordered lists of decisions made by agents along the paths in the network. Each propagated decision heads such a history. For instance the c-history propagated by A_i to any agent in $N(A_i)$ that performs T_l is a list of the form [$(A_i{:}T_k,\ A_j{:}T_l,\ \langle o_{i:k}, o_{j:l}\rangle)$ $|L$], where L is either an empty c-history or the c-history that has been propagated to A_i concerning T_k. By propagating c-histories, agents can detect cycles and take advantage of the transitivity of options' consistency, detecting positive/negative feedback to their decisions.

Specifically, an agent A_i detects a cycle by inspecting in a received c-history the most recent item originated by itself: Given a cycle $(A_1 \rightarrow A_2 \rightarrow ...A_{(n-1)} \rightarrow A_1)$, then for each decision $(A_1{:}T_k,\ A_2{:}T_l,\ \langle o_{1:k}, o_{2:l}\rangle)$ for the tasks T_k and T_l heading a c-history from A_1 to A_2, the originator must get a decision $(A_{(n-1)}{:}T_l,\ A_1{:}T_k,\ \langle o_{(n-1):k}, o_{1:k}\rangle)$ from the last agent $A_{(n-1)}$ in the cycle, s.t. $o_{(n-1):k}$ and $o_{1:k}$ are assumed by A_1 to be consistent, i.e. $(o_{(n-1):k},\ o_{1:k}) \in \sim_i$. In such a case the agent A_1 counts a *positive feedback* for its decision. In case there is a cycle but the forwarded decision is not assumed (according to A_i) consistent to $o_{1:k}$, , then there are one or more correspondences or decisions through the path that result to disagreements. In this case, the agent A_1 counts a *negative feedback* for its decision. It must be noticed that disagreements may still exist along a path when the agent A_1 gets positive feedback, but several decisions along the path compensate "errors". These cases are detected by the other agents, as the c-history propagates in the network. To make the computations more efficient and in order to synchronise agents' decision making, we consider that c-histories can be propagated up to 3 hops with repetitions: This means that given two peers A_i and A_j, any c-history starting from A_i (1st hop) shall be returned to this agent with the decision of A_j (2nd hop), and will return later to A_j with the new decision of A_i (3rd hop). In the last hop the agent A_i may revise its decision by considering also the feedback received from A_j, in conjunction with feedback from any other peer.

[3]It must be noticed that only one consistent option may be suggested per strategy. This may seem a restriction, but allows agents to receive precise suggestions w.r.t the strategies chosen.

3.3 Social Q-learning methods

$Q-$functions, or action-value functions, represent the future discounted reward for a state s when deciding on a specific strategy str for that state and behaving optimally from then on. The optimal policy for the agents in state s is to jointly make the choice $argmax_c Q^*(s, str)$ that maximizes the expected future discounted reward.

The next paragraphs describe three distributed Q-learning methods considering that agents do not know the transition and reward model (model-free methods) and interact concurrently with all their peers. All variants assume that agents propagate their decisions to peers, as explained in section 3.2.

Collaborative Reinforcement Learners (Colab-RL): This is the agent-based update sparse cooperative edge-based Q-learning method proposed in [12]. Given two peer agents performing their tasks,$[A_i{:}T_k \bowtie A_j{:}T_l]$, the $Q-$function is denoted succinctly Q([$i{:}k \bowtie j{:}l$],**s**,**str**), where **s** denotes the state variables related to the two agents performing their corresponding tasks (i.e. $s_{[i:k \bowtie \cdot:l]}$ and $s_{[j:l \bowtie \cdot:k]}$), and **str** denotes the strategies chosen by the two agents (i.e. $str_{i:k}$ and $str_{j:l}$). The sum of all these edge-specific $Q-$functions defines the global $Q-$function.

The update function is as follows:

$$Q([i{:}k \bowtie j{:}l], \mathbf{s}, \mathbf{str}) = Q([i{:}k \bowtie j{:}l], \mathbf{s}, \mathbf{str}) + \alpha \sum \frac{(Rwd_{[x:y \bowtie \cdot:z]}(sv_x, str_{x:y}) + \delta Q_{x:y}(sv'_x, str*_{x:y}) - Q_{x:y}(sv_x, str_{x:y}))}{|N(A_x)|}$$

where, the summation is for any $x : y \in \{i{:}k, j{:}m\}$. In case $x{:}y$ is $i{:}k$, then $z = l$, else $z = k$. Also, sv_x is the state variable $s_{[x:y \bowtie \cdot:z]}$.

The local Q-function for $A_i{:}T_k$ is defined to be the summation of half the value of all local functions Q($i{:}k \bowtie j{:}l$, **s**, **str**) for any $A_j \in N(A_i)$ performing task $T_l \in SocialContext(A_i)$: This is so, give that each local function considers two agents. Formally,

$$Q_{i:k}(s_i, str_{i:k}) = \frac{1}{2} \sum_{j:l, A_j \in N(A_i)} Q([i{:}k \bowtie j{:}l], \mathbf{s}, \mathbf{str}).$$

*Independent Reinforcement Learners:*The local function Q_i for an independent learner A_i is defined as a linear combination of all contributions from its neighbourhood, for any

task T_k assigned to A_i and performed jointly to tasks T_l:
$Q_i(s_i, str_i) = \sum_{T_k} \sum_{j:l, A_j \in N(A_i)} Q([i{:}k \bowtie j{:}l], s_i, str_{i:k})$,
where s_i is the local state of agent A_i and str_i its local strategy.
Each $Q([i{:}k \bowtie j{:}l], s_i, str_{i:k})$ is updated as follows:

$$Q([i{:}k \bowtie j{:}l], s_i, str_{i:k}) = Q([i{:}k \bowtie j{:}l], s_i, str_{i:k}) + \alpha[Rwd(s_i, str_{i:k}) + \delta * max_{str'_{i:k}} Q([i{:}k \bowtie j{:}l], s'_i, str'_{i:k}) - Q([i{:}k \bowtie j{:}l], s_i, str_{i:k})]$$

$Rwd(s_i, str_{i:k})$ does not follow the notation introduced, since we consider alternative ways of computing it. This method is in contrast to the collaborative approach introduced earlier since it computes values for strategies and states of individuals, considering only local states and strategies, This is also in contrast to the approach of Coordinated Reinforcement Learning model proposed in [11] since that model needs to know the maximising joint action in the next state, the associated maximal expected future return, and needs to estimate the global Q-value in the global state. It also considers society's global reward.

Focusing on agents' social context, we compute the local reward of agents based on their knowledge about the strategies proposed by acquaintances, as already specified above. Thus, in this case
$Rwd(s_i, str_{i:k}) = Rwd_{[i:k \bowtie \cdot:l]}(sv_i, str_{i:k})$,
where sv_i is again the state variable $s_{[i:k \bowtie \cdot:l]}$.

This model is also in contrast to the independent learners proposed in [6], since update functions consider agents' local states' only. This is a social learning method since agents consider the feedback received by others and it assumes that agents get their own payoff depending on the joint strategy chosen. We call this "individual & local learner model" and indicated as *Indl-RL*.

As an alternative to Indl-RL, the reward can be affected by the likelihood of peers to select any strategy. In this case the feedback is not computed as specified in formula (2). Actually, in this alternative an agent A_i does not consider the last messages received from peers to count the positive/negative decisions of peers (which may be myopic in some cases), but counts the likelihood to get a positive or a negative feedback from any peer based on the decisions reported by that peer: The strategy with the higher likelihood per peer is considered to be the most probable decision of that peer. In highly dynamic settings with many options per task (assuming that agents need to update their decisions frequently) this may not be a proper method, since the likelihood estimation may need a large number of samples to change, while agents may need to be more reactive. It must be noticed that agents may compute the likelihood of other strategies incrementally, requiring no memory. We refer to this last method as "independent likelihood estimation model" and indicated as *Indep-RL*, .

4. EXPERIMENTAL RESULTS

We have performed simulations using the three social learning methods presented in small-world networks that have been constructed using the Watts-Strogatz model (W) [25], and scale-free networks constructed using the Albert Barabási (B) model [2]. Given that results in both types of networks have not significant differences, and given that we are interested on operational complexity, we present results only for scale-free networks, varying the average number of neighbours (ANN) per agent in $\{4, 10, 20\}$ and the number of options per task in $\{2, 3, 4\}$. Each case is denoted by $X\_|N|\_ANN\_O$, where X is the network construction model (e.g. $B\_100\_10\_2$ for a scale-free network of 100 agents, with 10 neighbours per peer in average, and with 2 options per task). The society tasks $\mathcal{T}$ are 4 (e.g a,b,c,d) and each agent is randomly assigned 2 tasks in average satisfying the following constraints. One of these tasks (e.g. a) has been assigned in 10% of the population, while one of the other tasks (e.g. b) is assigned to an extra agent (e.g. the 101st agent in a network of 100 agents) and it is connected to agents performing a specific task (e.g. c). This models settings where only few in the population are able to perform specific tasks, while there are social tasks that need the coordination of numerous agents. In all methods the payoff $Profit$ for positive feedback and satisfaction of constraints is equal to 3, while the penalty $Penalty$ is equal to -5. Considering the reward, as already said, the feedback factor b is 10 times greater that the utility factor a, i.e. $b = 10a$. The reason for this is that, for agents to reach agreements, they need to consider peers' feedback to be very important, while also they consider compatibility constraints among their strategies. In cases where there are two options per task (e.g. p and np), each combination of tasks x, y may be associated with any of the following 2×2 payoff matrices given below.

	$p_{\cdot:x}$	$np_{\cdot:x}$
$p_{\cdot:y}$	-1,-1	3,2
$np_{\cdot:y}$	2,3	-1,-1

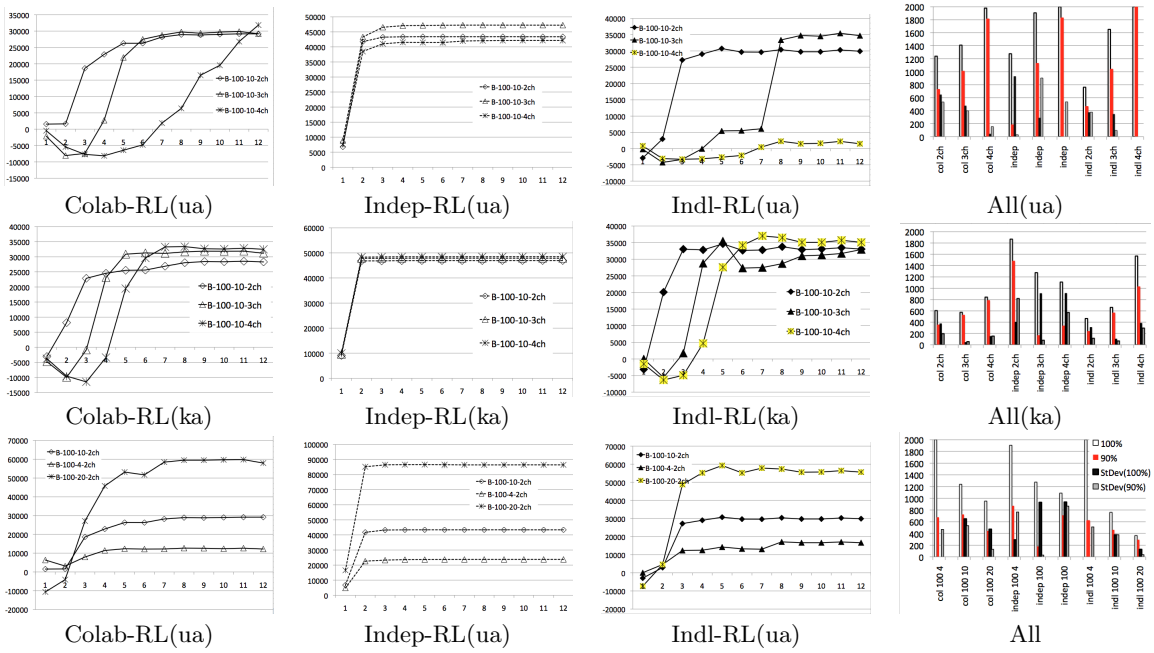
	$p_{\cdot:x}$	$np_{\cdot:x}$
$p_{\cdot:y}$	3,3	-1,-1
$np_{\cdot:y}$	-1,-1	3,3

In cases with more options, matrices are extended resulting to games of similar form: It must be noticed that agents play different types of games while performing joint tasks.

Learning methods use an exploration function, counting the number of times each decision has been made. An epoch comprises an exploration followed by a pure exploitation period. Aiming at effectiveness, we present results for 12 epochs (i.e. 2000 rounds) and measure the efficacy of the methods to converge until epoch 12. We measure convergence for 100% and for 90% of the agents. The convergence rule is that the required percentage of agents has reached agreement without violating any constraint in 10 subsequent rounds during an exploitation period. Each experiment has been performed 20 independent times and we present results by averaging the results recorded by these independent experiments.

Table 1 shows the results of methods in different settings and for different percentages of converging agents. The x-axis shows the epoch numbers and the y-axis in the line charts shows the total payoff received by the agents: The first (respectively, second) raw shows results for different number of options in B_100_10 networks with an unknown consistency relation (ua) (respectively, ka when it is known), according to the problem specification (respectively, contrary to the problem specification). The third raw shows how different methods are affected when the average number of peers per agent varies. These are again B_100_10 networks with two choices per task. The last column chart in each row shows the average convergence round (y-axis) per method, for the different cases considered in the corresponding raw, both for 100% and for 90% convergence, also with the standard deviation of the convergence rounds (denoted StDev) recorded. When the average round recored

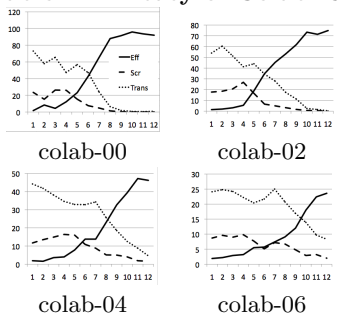
Table 1: Experimental results for all methods



is 2000, this means that the corresponding method did not manage to converge until epoch 12.

Results show that the problem considered in this article is harder than in the case where the consistency relation between options is known to agents: As shown in the second raw of Table 1, all methods manage to converge to agreed conversions until epoch 12. Methods converge in an early epoch, especially when agents have a small number of options to consider per task. Among them the Colab-RL method is more effective, even in cases of increased complexity, for 100% of the agents with low standard deviation. The Indep-RL method converges very effectively for 90% of the agents, but not that effectively for 100% of the agents, since the standard deviation for Indep-RL is high. The second column of Table 2 shows that indeed, Indep-RL manages to "climb" very fast to a maximum payoff for the society in all cases. However, as we will show, Indep-RL reaches a local maximum early enough. In contrast to that, when the consistency relation among options is unknown to agents, and as the complexity of the social setting increases (by increasing the number of options available per task), as shown in the first raw of Table 1, all methods fail to reach 100% convergence until epoch 12. Indl-RL fails to converge in case agents have 4 options per task. Indep-RL either fails to converge, or in cases where it can converge the standard deviation is high. Colab-RL on the other hand does not converge as effectively as Indl-RL for 2 options per task, but its effectiveness increases as the complexity increases, especially for 90% of the agents with 4 options. The last raw of table 1 shows the convergence of the methods in cases where agents have 2 options per task, but the number of acquaintances increase: Recall that any agent has to perform every task jointly with its peers. In this case Indep-RL has very high standard deviation in all cases, or it fails to achieve convergence, while Colab-RL is slightly better than Indl-RL as the complexity increases.

Table 2: Efficacy of Colab-RL



Delving into the details of methods' efficacy, we classify the states of agents depending on the feedback received from peers, into efficient (denoted by "Eff"), scrambled (denoted by "Scr") and transient (denoted by "Trans"), similarly to [19]. In efficient states agents receive positive feedback messages that are are at least 5 times more than the the negative ones. This means that in networks of 10 acquaintances per agent in average, an agent gets in average 2 negative feedback messages, and thus may be close to reaching agreement. In scrambled states the number of positive messages are more or less equal to the negative ones. Precisely, positives are ± 1.5 times as many as the negatives. The rest of the states are classified as transient. Furthermore, since all methods require communication between peers for monitoring peers' strategies, to test the tolerance of the methods in settings where agents have limited monitoring abilities, we consider that messages are received with a probability, which varies in $\{0, 0.2, 0.4, 0.6\}$. Average results from 20 independent experiments are reported in Tables 2, 3 and 4.

Each agent has 2 options per task. As it seems, Colab-RL manages to decrease the scrambled and transient states

Table 3: Efficacy of Indep-RL

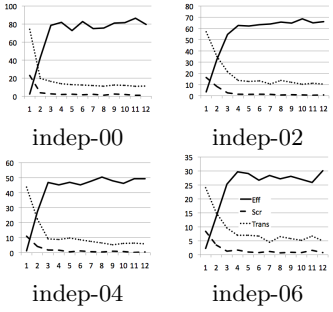
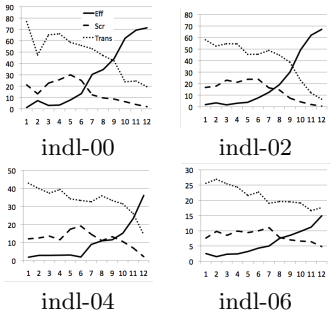


Table 4: Efficacy of Indl-RL



very effectively, even when the probability is 0.4. Even for probability 0.2 the number of these states is near to zero (in average below 1). On the other hand, Indep-RL has a stable behaviour for different probabilities, but it fails to reduce transient states, and scrambled ones are very low (near to 5 in average). Again, Indep-RL shows steep increase of efficient states early enough, but clearly now we see that it reaches local maxima. This however may serve as a good solution if we require a percentage of agents less or equal than 90% to converge. Indl-RL fails as well to reduce transient states effectively. Surprisingly, when the probability is 0.2 it demonstrates an effective reduction of scrambled and transient states before epoch 12, but this does not happen in other cases.

5. RELATED WORK

Early approaches towards learning norms or conventions either involve two agents iteratively playing a stage game towards reaching a preferred equilibrium, or models where the reward of each individual agent depends on the joint action of *all* the other agents in the population. Other approaches consider that agents learn by iteratively interacting with a single opponent from the population [17] [19], also considering the distance between agents [13], or by interactive repeatedly with randomly chosen neighbours [1]. In contrast to this, in [26] the communication between agents is physically constrained and agents interact and learn with all their neighbors. In these works agents play a single role at each time step. We rather consider cases where agents perform multiple tasks jointly with peers, simultaneously.

Concerning the effectiveness of reinforcement learning methods, Shoham and Tennenholtz [18] proposed a reinforcement learning approach using the Highest Cumulative Reward rule which depends on the memory size of agents. The effects of memory and history of agents' past actions have also been considered by Villatoro et al [20], [21]. Sen et al [17] [1] studied the effectiveness of reinforcement methods also considering the influence of the population size, of the possible actions, the existence of different types of learners in the population, as well as the underlying network topology of agents [16]. In [26] authors have proposed a learning method where each agent, at each time step interacts with all its acquaintances simultaneously and use ensemble learning methods to compute a final strategy.

Studies (e.g. [13], [17], [26]), have shown that Q-learners are competent to learners using for instance WoLF [5], Fictitious Play [9], Highest Cumulative Reward -based [18] models. Based on these conclusions and going beyond the state of the art, this work proposes social Q-learning methods, according to which agents interact with all of their acquaintances, considering their tasks in their social contexts, w.r.t. operational constraints. Similarly to [24], in this work agents have to learn the structure of a coordination game. Also, while [24] considers noisy payoffs and cases where agents have either perfect or limited monitoring abilities, this article considers that agents have limited interaction and monitoring abilities. Similarly, the approach described in [4] assumes that agents' actions are not directly observable by others and generalise fictitious play using likelihood estimates. In our case, we use likelihood estimates in the problem proposed as an alternative for estimating feedback.

6. CONCLUSIONS AND FURTHER WORK

This article investigates the effectiveness of computing conventions using social, distributed reinforcement learning methods in settings of increased complexity. The computation of conventions is done via reaching agreements in agents' social context, via interactions with acquaintances. The formulated methods support agents to reconcile and decide on the simultaneous performance of multiple tasks in their social contexts, jointly with peers with limited knowledge on options available and with limited monitoring abilities. The article formalises the generic problem where collaborative agents aim to coordinate with peers in settings with such an increased operational complexity, considering different types of constraints for the performance of tasks.

Experimental results show that Col-RL and Indl-RL are more effective than Indep-RL as the operational complexity of the setting increases. Furthermore, their effectiveness decreases as the complexity of the options available to agents increase, while it increases when the average number of acquaintances per agent increases. This is due to the exploitation of decision lists propagated among peers. Col-RL also proves to be tolerant in settings where agents have limited monitoring abilities.

Further work concerns investigating (a) the effectiveness of hierarchical reinforcement learning techniques [3] for computing hierarchical policies in cases where hierarchical tasks have to be performed; (b) the tolerance of the methods to different payoffs of performing joined tasks, as well as to different exploration-exploitation schemes, and (c) societies with different types of learners.

Acknowledgments

This work has been partly supported by the University of Piraeus Research Center.

REFERENCES

[1] S. Airiau, S. Sen, and D. Villatoro. Emergence of conventions through social learning - heterogeneous learners in complex networks. *Autonomous Agents and Multi-Agent Systems*, 28(5):779–804, 2014.

[2] R. Albert and A. lászlŮ Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys*, 2002.

[3] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(1-2):41–77, Jan. 2003.

[4] C. Boutilier. Learning conventions in multiagent stochastic domains using likelihood estimates. In *UAI '96: Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence, Reed College, Portland, Oregon, USA, August 1-4, 1996*, pages 106–114, 1996.

[5] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002.

[6] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, pages 746–752, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

[7] J. Delgado. Emergence of social conventions in complex networks. *Artif. Intell.*, 141(1/2):171–185, 2002.

[8] J. Epstein. Learning to be thoughtless: Social norms and individual computation. *Computational Economics*, 18(1):9–24, 2001.

[9] D. Fudenberg and D. Levine. *The theory in learning in games*. The MIT Press, 1998.

[10] C. E. Guestrin. *Planning Under Uncertainty in Complex Structured Environments*. PhD thesis, Stanford, CA, USA, 2003. AAI3104233.

[11] C. G. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *In Proceedings of the ICML-2002 The Nineteenth International Conference on Machine Learning*, pages 227–234, 2002.

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

[13] P. Mukherjee, S. Sen, and S. Airiau. Norm emergence under constrained interactions in diverse societies. In L. Padgham, D. C. Parkes, J. P. M§ller, and S. Parsons, editors, *AAMAS (2)*, pages 779–786. IFAAMAS, 2008.

[14] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[15] B. T. R. Savarimuthu. Norm learning in multi-agent societies. In *Information Science Discussion Papers Series No. 2011/05, Retrieved from http://hdl.handle.net/10523/1690*. 2011.

[16] O. Sen and S. Sen. Effects of social network topology and options on norm emergence. In J. Padget, A. Artikis, W. Vasconcelos, K. Stathis, V. da Silva, E. Matson, and A. Polleres, editors, *Coordination, Organizations, Institutions and Norms in Agent Systems V*, volume 6069 of *Lecture Notes in Computer Science*, pages 211–222. Springer Berlin Heidelberg, 2010.

[17] S. Sen and S. Airiau. Emergence of norms through social learning. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1507–1512, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[18] Y. Shoham and M. Tennenholtz. On the emergence of social conventions: Modeling, analysis, and simulations. *Artif. Intell.*, 94(1-2):139–166, July 1997.

[19] T. Sugawara. Emergence of conventions for efficiently resolving conflicts in complex networks. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Warsaw, Poland, August 11-14, 2014 - Volume III*, pages 222–229, 2014.

[20] D. Villatoro, J. Sabater-Mir, and S. Sen. Social instruments for robust convention emergence. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, pages 420–425. AAAI Press, 2011.

[21] D. Villatoro, S. Sen, and J. Sabater-Mir. Topology and memory effect on convention emergence. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 02*, WI-IAT '09, pages 233–240, Washington, DC, USA, 2009. IEEE Computer Society.

[22] G. Vouros. Decentralized semantic coordination via belief propagation. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 1207–1208, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[23] G. A. Vouros. *The Emergence of Norms via Contextual Agreements in Open Societies*, pages 185–201. Springer International Publishing, Cham, 2015.

[24] X. F. Wang and T. Sandholm. Learning near-pareto-optimal conventions in polynomial time. In *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*, pages 863–870, 2003.

[25] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, 06 1998.

[26] C. Yu, M. Zhang, F. Ren, and X. Luo. Emergence of social norms through collective learning in networked agent societies. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 475–482, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.