Auton Agent Multi-Agent Syst (2014) 28:749–778
DOI 10.1007/s10458-013-9240-2

A decentralized approach for convention emergence in multi-agent systems

Mihail Mihaylov · Karl Tuyls · Ann Nowé

Published online: 11 October 2013

Abstract The field of convention emergence studies how agents involved in repeated coordination games can reach consensus through only local interactions. The literature on this topic is vast and is motivated by human societies, mainly addressing coordination problems between human agents, such as who gets to redial after a dropped telephone call. In contrast, real-world engineering problems, such as coordination in wireless sensor networks, involve agents with limited resources and knowledge and thus pose certain restrictions on the complexity of the coordination mechanisms. Due to these restrictions, strategies proposed for human coordination may not be suitable for engineering applications and need to be further explored in the context of real-world application domains. In this article we take the role of designers of large decentralized multi-agent systems. We investigate the factors that speed up the convergence process of agents arranged in different static and dynamic topologies and under different interaction models, typical for engineering applications. We also study coordination problems both under partial observability and in the presence of faults (or noise). The main contributions of this article are that we propose an approach for emergent coordination, motivated by highly constrained devices, such as wireless nodes and swarm bots, in the absence of a central entity and perform extensive theoretical and empirical studies. Our approach is called Win-Stay Lose-probabilistic-Shift, generalizing two well-known strategies in game theory that have been applied in other domains. We demonstrate that our approach performs well in different settings under limited information and imposes minimal system requirements, due to its simplicity. Moreover, our technique outperforms state-of-the-art coordination mechanisms, guarantees full convergence in any topology and has the property that all convention states are absorbing.

M. Mihaylov (✉) · A. Nowé
Vrije Universiteit Brussel, Pleinlaan 2, Brussels, Belgium
e-mail: mmihaylo@vub.ac.be

K. Tuyls
Maastricht University, Sint Servaasklooster 39, Maastricht, The Netherlands

Present Address:
K. Tuyls
University of Liverpool, Ashton Street, Liverpool L69 3BX, UK

Keywords Coordination games · Convention emergence · Decentralized systems

1 Introduction

Coordination problems are commonly studied through repeated interactions between members of an artificial society [2,25,36]. Such studies can be classified as either descriptive—attempting to construct a model of how coordination emerges in human populations (e.g. in the evolution of language [38]), or prescriptive—proposing strategies that lead to successful coordination in the absence of a central authority [37]. The research in both categories is motivated by human societies, where members of the population are involved in symmetric asynchronous pairwise interactions.

Although the research in the prescriptive category proposes numerous techniques for coordinating the actions of agents, it focuses mainly on human societies and does not directly address engineering applications. Real-world engineering problems pose certain restrictions on the complexity of the coordination mechanisms and the information available to agents. For example, wireless nodes are typically small devices with limited memory and processing capabilities and therefore cannot execute complex algorithms or store large quantities of data. Moreover, agent interactions are influenced by the application domain, which may also restrict whether agents are aware at all that an interaction is taking place. For example, a wireless transmission between two devices affects all other wireless devices in range, thus interactions cannot be considered as pairwise, as commonly assumed in literature, but are multi-player instead. Furthermore, if the radio of an intended receiver is switched off, damaged, or currently in communication with another agent, it cannot detect that another transmitter is trying to send information and therefore does not update its action. The intended receiver is simply unaware of its involvement in an interaction with the transmitter, while the latter agent is aware of the "game" that is taking place and can update its action. Another difference with human coordination that is rarely considered in literature is the timing of the action updates. While pairwise asynchronous updates are commonly encountered in human populations (e.g. in telephone calls), computer agents may follow a communication protocol (e.g. Time Division Multiple Access type protocol in wireless sensor networks), where all agents synchronously interact and update their strategies at every iteration (or time step). To summarize, coordination problems between computer devices may significantly differ from the models assumed in the literature on human coordination. Computer agents may interact pairwise, or multi-player, using synchronous or asynchronous updates and may not always be aware that an interaction is taking place. In other words, systems comprised of computer agents may exhibit different properties, depending on the agents themselves and the application domain, resulting in different system dynamics, as it will become evident later on in this article. Therefore, strategies proposed for coordination problems between human agents cannot be directly applied in engineering applications and need to be further explored in the context of real-world application domains.

Similarly to biological systems, many computer systems are comprised of multiple entities (or agents) with common objectives. Individual agents have restricted capabilities, but the group as a whole is capable of executing more complex tasks than a single agent can perform. Coordination challenges, faced by such multi-agent systems, are inherently decentralized. Central control is simply unavailable and costly to set up in problems such as wireless sensor nodes monitoring habitats, or robot swarms exploring unfamiliar terrains. In these settings individual agents are simply unable to fulfill the design objectives of the whole system when acting on their own. In those multi-agent systems, agents need to efficiently coordinate their behavior in a decentralized and self-organizing way in order to achieve their common, but

complex goals. Therefore it is the task of the system designer to implement efficient mechanisms that enable the emergence of decentralized coordination between highly constrained agents. In this article we take the role of designers of large decentralized systems and investigate the following problem, which motivates our research: *How can the designer of a decentralized system, imposing minimal system requirements and implementation overhead, enable the efficient coordination of highly constrained agents, based only on local interactions and limited knowledge?* Thus our research falls into the prescriptive category of the coordination game literature and is motivated by engineering applications, such as swarm robotics and wireless sensor networks.

Lewis [25] describes the coordination problem as a game in which agents can realize mutual gains by selecting the same action in the presence of several alternatives. In particular, we are interested in *pure* (or common interest) coordination games. The agents care little on which of the available actions they will coordinate, as long as all agents select the same action [33]. At a given iteration a pair of agents meet, interact and update their strategies with the aim to all eventually arrive at a convention. Most generally, a convention is a solution to a coordination problem, where agents have reached an agreement and thus realize mutual gains. For example, if a telephone call between two participants gets unexpectedly dropped, there is no central authority to select who should call back. After repeated encounters agents may learn that the connection can be quickly re-established if, for example the original caller redials while the other agent waits. Such behavior, if adopted by all members of a given population, becomes a convention. Note that although the literature on human coordination assumes common interest, human agents are typically self-interested and do have individual preferences. In this article we are not considering human coordination, which has already been extensively studied in literature [1,25,37,41]. Instead, we focus on the coordination challenge of computer agents in large cooperative multi-agent systems. We examine the pure coordination game in an abstract form and assume that all devices in the system are owned by the same user and therefore agents are not self-interested. The coordination challenge in such systems can take any form. For example, a fleet of quadrotors (or mini-helicopters) guarding a military object may need to coordinate on neutralizing one intruder at a time for a higher chance of success, rather than spreading their forces on multiple targets; wireless nodes need to coordinate on reserving the same broadcast channel for control messages; and so on.

In this article we propose a decentralized approach for convention emergence and compare it to several other techniques described in literature. Similar to reinforcement learning scenarios, each agent receives a payoff from its interaction with others and uses it to update its own action, which then influences the payoff experienced by the agent in the next iteration. Agents keep their action if they experience no conflict and probabilistically select a different one if they encounter conflicts.

The remainder of this article is organized as follows. In the next section we situate our work on convention emergence in the context of related work and then outline our contributions in Sect. 3. We study in detail the coordination game that our agents are involved in and the underlying interaction models in Sect. 4. We describe our approach in Sect. 5 and investigate its performance in different settings in Sect. 6. We present our conclusions in Sect. 7 and provide some directions for future work.

2 Related work

In this section we examine the related work by grouping it according to a number of characteristics. Some of the main features we explore are topology type, memory size, interaction

model and convergence threshold. These characteristics will allow us to compare the different settings used in the literature on convention emergence.

The related work presented below considers populations of artificial agents as well as human populations, in which players are not self-interested and all aim towards the same goal. We use the same assumption of altruistic agents in this article, but we restrict our attention to computer agents.

One of the earliest and most influential works on the study of conventions is that of Lewis [25]. He explores the emergence of conventions and the evolution of language in signaling games. Later, Axelrod [2] investigates the factors that speed up convention emergence and the conditions under which a convention remains stable. Young [44] studies stochastically stable equilibria in coordination games where agents can occasionally explore, or make mistakes. These authors, as well as others [11] consider only a fully connected **network** of players where each agent can potentially interact with any other agent with the same probability. Results from these studies are still relevant to engineering applications if we can assume that all devices are within communication range of each other, e.g. a fleet of quadrotor swarmbots flying in a close formation. However, we cannot always assume that any two agents in a multi-agent system can interact with each other. Large obstacles, such as buildings or hills, may prevent direct communication, or agents may simply not have the communication power to reach all others. In these settings we say that agent interaction obeys a certain interaction model. In that regard, the work of Kittock [21] is the first to explore the influence of a restrictive interaction model on the evolution of the system. He investigates the performance of agents in different interaction graphs (or topologies). Similar studies investigate the speed of convention emergence in scale-free networks [12] and whether these networks facilitate agreement between agents in continuous strategies [17]. Restrictive topologies are indeed more general than fully connected topologies and often represent more realistic scenarios.

A large body of research has studied convention emergence as one-to-one **interactions** between randomly selected individuals of the population [4,8,21,37], where agents typically choose between only 2 actions. Little work has focused on games with more than 2 actions and on multi-player interactions where an agent can meet a variable number of other players in a single game. Notable exceptions are Delgado et al. [12] and Villatoro et al. [42]. Note that some authors use the term "action" to refer to the coordination alternatives agents need to choose from, while others prefer the term "state". In this article we use the former term in order to distinguish its use from the term "system state", which stands for the combination of actions of all agents in the system.

Shoham and Tennenholtz [36] introduced the highest cumulative reward (HCR) update rule that lets agents select the most successful action in the last l iterations. Thus, the parameter l is the number of previous interactions of the agent, or its **memory**. In the latter work, the authors assume that agents have access to either the entire history of plays (i.e., $l = \infty$), or just a limited memory window. Similarly, Bendor et al. [5] and Karandikar et al. [18] present algorithms that use the history of past plays to determine whether an action falls short of a predetermined threshold (also called aspiration level), which varies based on the payoffs of previous interactions. The assumption of unlimited history of interactions is often unrealistic, e.g. in wireless sensor networks, where nodes have limited memory capabilities. Although HCR with limited memory converges relatively quickly in fully connected topologies, its convergence is not guaranteed in restrictive topologies, such as a ring or a scale-free graph. Selecting the most successful action in the last l iterations introduces what Villatoro et al. refer to as "the frontier effect". A number of agents at the border between two groups, who each has converged to a different action, experience conflicts, but cannot resolve them, since the most successful action for each agent is reinforced by its other neighbors in the graph. In fact,

Barrett and Zollman [4] investigate how forgetting past interactions may help agents reach conventions faster in the evolution of language. They conclude that forgetting helps move agents away from suboptimal equilibria (such as the frontier effect in coordination games) and therefore increases the probability of evolving an optimal language in signaling games. Bowling and Veloso [6] introduce the WoLF-PHC algorithm (win or learn fast-policy hill climbing), where agents, instead of remembering the last l actions, keep only an expectation of the payoff of each action at each state. WoLF-PHC is guaranteed to converge to the Nash equilibrium in 2-player 2-action games, but such guarantees are not shown for games with more than 2 agents. Sen and Airiau [35] apply WoLF-PHC in a coordination game with multiple actions and show that conventions can emerge in a fully connected topology. However, they too observe the frontier effect when groups of agents interact infrequently. Therefore, their approach is not guaranteed to converge in the settings studied in this paper.

Most of the literature on convention emergence assumes that a convention is reached when at least 90 % of the population select the same action [12,21]. However, Villatoro et al. [42] argue that a **threshold** of 90 % is not sufficient to say that agents' behavior has converged. As we will see in Sect. 6 unless 100 % of the population selects the same action, there is always the possibility that the majority agents may learn, over time, to select a different action and therefore drift away from that convention and arrive at a different one. Moreover, if any group of agents arrive at a different (sub-)convention than the rest of the population, the agents on the border between the different groups will experience conflicts and thus incur costs. In a wireless sensor network, for example, if some nodes are in conflict with others, they will deplete their batteries faster than the rest of the network, drastically lowering the overall network lifetime. For engineering applications we require that in order for a coordination problem to be solved there may not exist any sub-coalitions, i.e. groups of agents whose actions differ from those of other groups. For the remainder of this article we say that the population has reached a convention, or that the individual's behavior has converged, if and only if *all* agents have learned to select the same action, regardless which, as agents are not self-interested.

Due to the stochastic action-selection policy (ε-Greedy) of Villatoro et al. [42], even if 100 % of the population select the same action at some moment in time, agents may still escape the convention in the next step and, over time, form new one. We therefore say that such conventions are not stable, i.e. the convention state is not absorbing. In contrast, our approach ensures that convention states are stable—once all agents select the same action, they will **never escape** this state, unless the system is disturbed, or new agents are added to the system with random initial actions. We name this approach Win-Stay Lose-probabilistic-Shift (WSLpS) and it shares characteristics with two related techniques proposed in literature. Barrett and Zollman [4] introduced the Win-Stay Lose-Randomize (WSLR) algorithm in the context of the evolution of language. WSLR is maximally forgetful, in the sense that it retains no history of past interactions, and outperforms two different reinforcement learning algorithms that use memory. The authors draw an analogy to a similar algorithm, named Win-Stay Lose-Shift (WSLS). The basic idea of the WSLS algorithm is that the agent will keep on selecting the same action, as long as its payoff is above a certain threshold level (also called aspiration level), and will change its action when the payoff drops below that level. It was first presented as Win-Stay Lose-Change by Kelley et al. [19] and later analyzed by Nowak and Sigmund [31] in the iterated Prisoner's Dilemma game. As we will see in Sect. 5, our action selection approach resembles, but outperforms, both WSLR and WSLS, and differs in the way agents select their action upon failure.

In Table 1 we summarize the contributions of other researchers according to the features presented in this section and situate our work in this domain.

Table 1 Summary of related work

Author(s)	Approach	Memory size	Topology type	Interaction model	Convergence threshold
Kittock	HCR	Limited, none	Full, restrictive	2-Player	90 %
Young	Adaptive play	Global, limited	Full	n-Player	Not reported
Shoham and Tennenholtz	HCR, EM	Full, limited	Full	2-Player	85–95 %
Delgado et al.	HCR, GSM	Limited	Restrictive	2-Player, n-player	90 %
Sen and Airiau	WoLF-PHC	Limited	Full	2-Player	Not reported
Urbano et al.	RFR, RSR	None	Full, restrictive	2-Player, n-player	90 %
Barrett and Zollman	ARP, smoothed-RL, WSLR	Full, limited, none	Full	2-Player	99 %
Villatoro et al.	Q-learning	Limited	Full, restrictive	2-Player, n-player	100 %, not stable
Mihaylov et al.	WSLpS	None	Full, restrictive	2-Player, n-player	100 %, stable

We survey in this article different types of pure coordination games where the objective of agents is to all learn to select the same action, e.g. agents in a robot swarm agreeing to execute the same task. A different type of common interest coordination problems is one where agents need to anti-coordinate instead [7,16,23]. In wireless communication, for example, neighboring devices need to transmit on different frequencies in order to avoid communication interference. Mihaylov et al. [28] applied the predecessor of WSLpS in such setting in an actual engineering scenario. The anti-coordination problem, thus, arises when multiple agents need to select actions, such that no two adjacent agents have the same action. Although pure coordination games and pure anti-coordination games are just two sides of the same coin [27], in this article we focus on the former in order to illustrate in more details the different settings that arise in coordination games between computer agents. We refer readers who are interested in the relation between these two types of convention emergence problems to Mihaylov [27]. Lastly, we consider here a homogeneous system where all agents are owned by one user and investigate how fast all agents can independently arrive at the same decision. In contrast, Franks et al. [15] study how a group of influencer agents can manipulate a system with heterogeneous ownership to reach a particular (or desired) convention in the context of language games.

We draw here a parallel to the literature on the emergence of cooperation [17,34]. It also studies how cooperation/coordination can emerge from local interactions, e.g. through gossiping [39]. However, each agent may employ a different learning mechanism, whose fitness is tested against that of others. In contrast, in the literature on coordination the system designer imposes the learning algorithm with the aim to coordinate the actions of individual agents. The literature on opinion dynamics, another field related to convention emergence, seeks mathematical rules to describe the evolution and changes of opinions in large social groups. As such the literature in this field falls in the descriptive class of studies. Castellano et al. [9] survey different models that describe the spreading of opinions in both finite and infinite populations. Typical models assume static topologies and suggest that agents directly impose their opinions on random agents in a fully connected topology. Although convergence is (asymptotically) guaranteed, it is not clear how these guarantees extend to dynamic topologies. Moreover, in restrictive topologies, e.g. in small-world networks, the voter model of spreading opinions also encounters the frontier effect in the form of a metastable state with

coexisting domains of opposite opinions. Another model that draws parallels to multi-player interactions studied here is the majority rule model, where a group of agents is selected at random and all agents in that group adopt the action of the majority. Other models have different assumptions on the information available to agents, e.g. agents may observe the opinions of indirect neighbors, and propose additional characteristics of agents, such as persuasiveness and supportiveness. A notion similar to imposing opinion is investigated by Urbano et al. [40], where they propose a strategy called Recruitment based on Force with Reinforcement. Strategies adopted by a large number of agents will have a higher force, thus higher probability to recruit weaker agents, who imitate the strategy of their stronger neighbors.

3 Summary of contributions

Many real-world scenarios can be modeled as abstract coordination games, where agents learn to coordinate through repeated interactions. In this article we study abstract pure coordination games played by computer agents in a multi-agent system. In most settings agents need to coordinate in the absence of a central authority. Moreover, due to the implied cost of miscoordination, e.g. between battery-powered devices, conventions need to be followed by all agents and need to emerge in as few interactions as possible.

Our main contributions are (1) to propose a decentralized approach for on-line coordination, (2) to analytically study its properties using the theory of Markov chains and (3) to perform an extensive empirical study analyzing the behavior of agents in a wide range of settings. Our approach is called Win-Stay Lose-probabilistic-Shift (WSLpS). The approach uses a parameter that defines the probability with which agents will change their action upon miscoordination. Since some devices, such as swarmbots, have limited memory capabilities, we require that learning approaches impose minimal requirements on the memory of agents. WSLpS requires *no memory* of previous interactions, given the current one, and drives agents to *full convergence*, i.e. 100 % of the agents reach a convention. Moreover, once convention is reached, agents will never change their actions and thus *never escape* the convention, unless the system is externally disrupted. To the best of our knowledge, WSLpS is the first coordination approach that (i) guarantees full convergence in any topology and (ii) has the property that all convention states are absorbing.

We investigate what factors can speed up the convergence process of agents arranged in different *topological configurations*, such as ring, scale-free, random and fully connected topologies. We also study the performance of WSLpS in dynamic topologies, where network links may rewire. Since interactions between individuals often incur certain cost (e.g. time, computational resources, etc.), it is necessary that agents reach a convention in the least number of interactions.

Forwarding a message in wireless sensor networks requires the coordination of at least two nodes. A transmission, however, may affect all nodes in range. In addition to *pairwise* interactions, that are explored in literature, we study the emergence of conventions as *multi-player* games where an agent interacts with possibly many other neighbors at the same time and obtains a single feedback from that interaction. This type of one-to-many encounters is rarely studied in literature, but often occurs in real-world applications, such as in wireless sensor networks.

Other assumptions in the literature on pure coordination are that games are symmetric, i.e. all agents in a game receive the same payoff, and that interactions are asynchronous, i.e. a subset of agents is selected to play a game at a given time. To account for some typical scenarios in engineering applications, in this article we also investigate settings where games

Table 2 Payoff matrix of the symmetric 2-player k-action pure coordination game. Each cell shows the payoff of agents i (row player) and j (column player), respectively

	a_j^1	a_j^2	$\cdots$	a_j^k
a_i^1	(1,1)	(0,0)	$\cdots$	(0,0)
a_i^2	(0,0)	(1,1)	$\cdots$	(0,0)
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
a_i^k	(0,0)	(0,0)	$\cdots$	(1,1)

are *asymmetric*, as agents cannot be assumed to be aware they participate in a game, and where interactions are *synchronous*, e.g. relying on a synchronous communication protocol. We also relax the assumption on common observability, where agents are assumed to see each other's actions, since *observation* is typically costly in computer systems or is even impossible in some cases. Lastly, we investigate system convergence using WSLpS in the presence of *faults*, i.e. if for some reason an agent is "stuck" with the same action and cannot change its choice. We demonstrate how the theorem of impossibility of consensus with one faulty process [14] does not always hold when agents are using the WSLpS approach.

4 Model

In Sect. 1 we outlined the concept of a coordination game and gave some examples of coordination problems. Here we will elaborate on the coordination game we consider, as presented in literature, together with the model of agent interaction that we use in our analysis and simulations.

4.1 Coordination game

For a proper comparison with other coordination algorithms, we use the same experimental setting as in the literature on coordination games [2,25,36]. Agents are involved in symmetric 2-player games with random individuals from the population, where both agents can observe each other's action when the interaction is taking place. The same one-on-one coordination game is played throughout the whole network between randomly selected **pairs** of neighboring agents. The goal of agents is to all learn to consistently select the same action, regardless which.

The game between two agents i and j at a given iteration is represented by a two-dimensional payoff matrix M, shown in Table 2. The action set of agent i is $\{a_i^1, a_i^2, \ldots, a_i^k\}$, and similarly, $\{a_j^1, a_j^2, \ldots, a_j^k\}$ is the action set of agent j. We assume here that the action sets of all players are identical, though in certain coordination problems agents may have different action sets. In addition, all payoffs are binary—actions either coincide (payoff of 1), or differ (payoff of 0).

The action of each agent is initialized uniformly random. Time t is continuous and interactions are **asynchronous**, such that at any given time interval $\triangle t$ the probability that more than one pair of agents interacts is infinitesimal. For this reason, and without loss of generality, we discretize time into time intervals, where an interval is defined by an interaction. Thus, at every time interval (or iteration), two randomly selected neighboring agents are paired in a game and receive payoff, as shown in Table 2. Since both agents receive the same payoff, the game is **symmetric**. The payoff for both agents is determined based on the combination of their actions, which is also called their joint action. Thus, the payoffs in the game matrix capture the incentive for agents to reach a convention.

Based on the obtained payoff, the two agents **simultaneously update** their actions. A classical game-theoretic assumption is that agents select actions in order to maximize their payoff. For this reason, rational players must choose an action based on their expectation of the play of others. However, agents are not able to see the actions of others *before* they interact, so as to anticipate the outcome of the game, neither can they see the newly selected action *after* each interaction, to align their choice. The common assumption in coordination games is that agents can only see each other's action *during* the interaction. As even this assumption is not realistic in some engineering applications, we will further relax it in Sect. 6.4, where agents cannot observe the actions of others at all.

After the two agents update their actions, a new random pair of agents is selected to play the pure coordination game, and so on. Thus, agents are involved in symmetric asynchronous pairwise interactions, where both agents are aware of the game and can observe each other's actions. For example, such a game can be played between swarmbots—at random times agents that are close to each other exchange information on the task they plan to accomplish, before attempting to connect with other agents. In Sect. 6.4 we will study other interaction models that are also typical for engineering applications.

It is generally assumed that there is no logical order of the actions of agents and thus they are not labeled in any specific way. For example, if at a given moment a swarm of quadrotors, guarding a military object, detects five vehicles coming from different directions, there is no common way to determine which is vehicle number 1, number 2, etc., so that the system designer can program the swarmbots in advance to select actions (i.e. to neutralize vehicles) in a given order, e.g. number 1 first, then number 2 and so on. A similar problem may arise even if there exists a logical order of actions. If a wireless sensor network designer programs all nodes to reserve wireless channel number 1 for control messages (instead of letting nodes coordinate) and it becomes unavailable due to interference, nodes could spend a lot of energy attempting to communicate on that frequency. Using a predetermined sequence of channels poses a similar problem, as agents cannot determine in a decentralized manner *when* all nodes should switch to the next channel in the programmed sequence. To avoid these trivial and often unrealistic settings, we require that the behavior of agents is independent of action names and their order. This assumption is common in coordination games and is known as action symmetry.

4.2 Interaction model

We consider a fixed population of N agents, arranged in a connected interaction graph (or topology). Vertices represent agents and a direct interaction between agents is allowed only if they are connected by an edge in the graph. Players who share an edge are called neighbors and all neighbors of a given player constitute its neighborhood. We first study static topologies, which are typical in wireless sensor network monitoring applications, and later in Sect. 6.5 we explore networks of dynamic connectivity, which are suitable to model the connectivity of mobile agents, such as agents in swarm robotics. We note here that agents are unaware of the identity (or names) of others. Thus, players cannot condition their action selection on agent names. This assumption, called agent symmetry, stems from a similar requirement we stated earlier—agents are affected only by *what* others are playing, i.e., their actions, and not by *whom* they are playing with, or their identity. The intuition behind this principle is that we cannot anticipate in advance which particular individuals will be involved in a coordination game, as the topology is not a priori known and may change. Even though information on the identities of other agents might be quite informative in some settings (e.g. a backbone

Algorithm 1 Main simulation process for the pure coordination problem

Input: $N \leftarrow$ number of agents,
$S \leftarrow$ type of topology,
$T_{max} \leftarrow$ maximum iterations
Output: iterations t until full convergence or T_{max}

$g \leftarrow initTopology(N, S)$
$a \leftarrow selectRandomActions$
$t \leftarrow 0$
repeat
  $i \leftarrow selectRandomAgent()$
  $j \leftarrow selectRandomNeighbor(i, g)$
  $p_i \leftarrow getPayoffFromGame(a_i, a_j)$
  $p_j \leftarrow getPayoffFromGame(a_j, a_i)$
  $a_i \leftarrow updateAction(p_i, a_i, a_j)$
  $a_j \leftarrow updateAction(p_j, a_j, a_i)$
  $t \leftarrow t + 1$
until $conventionReached(a)$ **OR** $t \geq T_{max}$
return t

node in a heterogeneous wireless sensor network), the role of agent identities is out of the scope of this article, as it is often done in the literature of coordination games [11,37,41].

In addition to a theoretical analysis, we study the iterated abstract coordination game in a simulation that proceeds as follows. At every iteration, one uniformly random pair of neighboring agents is selected to play the pure coordination game, outlined in Sect. 4.1. Both agents receive payoff, based on their actions and each one uses its action selection mechanism to update its action, which will be used in the next game they are selected to play. After that, the new iteration begins. Our performance criterion here is the number of iterations until full convergence is reached, or until T_{max} iterations have passed. This simulation process is detailed in Algorithm 1. We set T_{max} high enough to allow for a sufficient number of attempts to reach convention. Note that agents themselves are not aware of the global behavior of the population, i.e. the fact that a convention has been reached, and therefore they will continue playing the pure coordination game. Their action selection process must ensure that they do not leave the state of convention. We will see in Sect. 5.2 how our action selection algorithm ensures just that.

5 Win-Stay Lose-probabilistic-Shift approach

In the presence of several alternatives and aligned preferences agents need to rely only on repeated local interactions to reach a convention. Here we outline the learning process of agents when using our approach. After each interaction the participating agents use their payoff signal to decide whether to select the action of the neighbor or keep their action unchanged in the next iteration.

Intuitively, if an agent receives the maximum payoff from an interaction ("win"), it means that the other player in that given encounter has selected the same action. In that case it is reasonable that the agent will select the same action in the next interaction ("stay"). A low payoff, on the other hand ("lose"), indicates that the selected neighbor has picked different action and therefore each agent needs to possibly shift its action to the one of the neighbor in the next interaction ("probabilistic shift"). Whereas in the classic WSLS and WSLR agents will always change their action upon "lose", in our version we introduce a probability for

Algorithm 2 function *updateAction* for the pairwise interaction model

Input: payoff p_i and action a_i of agent i from the latest interaction as well as action a_j of agent j
Output: the new action a_i of agent i

1: $rnd \leftarrow generateRandomNumber(0, 1)$
2: $\Pi_i \leftarrow \max(\alpha - p_i, 0)$
3: **if** $rnd < \Pi_i$ **then**
4: $\quad a_i \leftarrow a_j$
5: **end if**
6: **return** a_i

"shift". This stochasticity is necessary to ensure that agents with conflicting actions will not constantly alternate their choices and thus will reach a convention faster. We will see in Sect. 6 that none of the above two algorithms can outperform ours in pure coordination games. Due to the probabilistic component, we name our action selection approach WSLpS.

The pseudo-code of our action selection approach (function *updateAction* from Algorithm 1) is presented in Algorithm 2. Agent i will keep its last action unchanged in the next iteration if it obtained a payoff of 1 ("win-stay"). If $p_i = 0$, on the other hand, agent i will adopt the action of agent j with probability α ("lose-probabilistic-shift"); with probability $1 - \alpha$ the agent will keep its action unchanged. In other words, α is the *shift probability* upon conflict. This parameter gives the probabilistic component of our WSLpS. A value of α close to 1 drives agents to alter their actions more often, while a value close to 0 makes them more "stubborn". Note that if $\alpha = 1$, our approach resembles the classic WSLS. Agents will always exchange their actions when they obtain a payoff of 0, which results in constant oscillation of actions especially in 2-action games. In WSLR, on the other hand, upon conflict each of the two agents will select an action at random. This behavior may cause the agent to select the same action as in the last iteration with probability $1/k$, where k is the number of actions. Therefore, in the 2-action case WSLpS resembles WSLR if $\alpha = 1/2$. If $\alpha = 0$, on the other hand, agents will never select a different action and therefore never reach a convention. For this reason we require that α lies in the interval (0,1]. The value of α is the same for all agents and it is fixed at the beginning of the simulation, i.e. at design time.

In essence, each agent participating in a coordination game has a probability of α to adopt the action of the agent with whom it interacts. As mentioned earlier, we assume here that both agents are aware that the interaction takes place and thus both agents update their actions. Since both agents perform an update after the interaction takes place, the value that gives the best chance for agents to select the same action is, intuitively, $\alpha = 0.5$. We will see in Sect. 6.4 that the best value of α is different when only one agent is aware of the game.

5.1 Properties of WSLpS

One important property of our WSLpS approach is that it is fully decentralized. The algorithm is run independently by each agent and updates the agent's action only based on local interactions. Agents need not be aware of distant players or their payoffs. Propagating such information in large networks can be costly or unreliable. Communication in wireless sensor networks, for example, is costly in terms of energy consumption and can also be unreliable due to external interferences or can be delayed in time. In addition, our algorithm does not rely on explicit message passing for coordination, as agents use only the outcome of actions to individually determine their next actions. In contrast, Farinelli et al. [13] present a decentralized coordination algorithm that requires agents to explicitly exchange messages to compute the global welfare of the system.

Another interesting property is that agents do not need to keep a history of (recent) interactions. Most coordination approaches proposed in the literature base agent's action selection on the history of past interactions [42,44]. In our case, and as in Sen and Airiau [35] and Mukherjee et al. [30], an agent selects an action based only on the current interaction and therefore requires no memory, apart from the one necessary to store the algorithm itself. Yet another desirable aspect of our action selection mechanism is that a convention can be reached in a finite number of iterations (see Theorem 2).

5.2 Markov chain analysis

In analogy to Brooks et al. [8], we too study the theoretical properties of our approach in order to provide certain guarantees that cannot be captured by pure empirical analysis. Note however, that although Brooks et al. [8] provide a proof of convergence, it is not clear whether convergence is guaranteed in finite time and that no absorbing loops exist. In our analysis we address these concerns.

In order to study the expected convergence time and the parameter α of our approach, we can represent our system as a Markov chain. In a network of N agents with k actions a system state s is an N-tuple that contains the action of each agent at a given iteration. The set of all N-tuples (or system states) constitutes the state space $S = \{s_1, s_2, \ldots, s_{k^N}\}$. We are interested in the probability π_{s_x,s_y} (or $\pi_{x,y}$ for short) of going from state s_x to s_y in one step. Thus, $\forall s_x, s_y \in S$, $\pi_{x,y}$ constitute the elements of the row-stochastic transition matrix $\mathscr{P}$. The transition matrix tells us the probability of transitioning between any two states in a single iteration. To compute the probability of going to a given state in any number of iterations from any starting state, we use the following theorem (see Kemeny and Snell [20] for proof).

Theorem 1 *Given a probability vector $\vec{u}$ representing the distribution of starting states and a transition matrix $\mathscr{P}$, the final probabilities $\vec{u}^{(t)}$ of arriving at each state after t iterations is:*

$$\vec{u}^{(t)} = \vec{u}\mathscr{P}^t \tag{1}$$

Thus, the yth entry in the vector $\vec{u}^{(t)}$ shows the probability that the Markov chain is in state s_y after t iterations, starting from an initial distribution $\vec{u}$. We are interested in arriving at those states s_y which are conventions, i.e. in which all agents select the same action. We use Eq. 1 to determine the probability of arriving at those states in a given number of iterations starting from any initial state. In a similar manner we can use $\mathscr{P}$ to determine the expected convergence time of the system.

To compute each element in $\mathscr{P}$, we need to determine the transition probability between each pair of system states for a single step. To aid the discussion, we will represent each index of $\mathscr{P}$ (i.e. each state in S) as an N-digit number with base k, the number of available actions. The first digit shows the action of the first agent, the second digit—the action of the second one and so on. Furthermore, we define the operator $\oplus$ that shows the binary "difference" between two states s_x and s_y, having a functionality similar to the XOR operator on binary numbers. Thus, $s_x \oplus s_y$ gives a vector $\vec{b}_{x,y}$ with N elements. The ith element of $\vec{b}_{x,y}$ is 1 if the ith digit of s_x differs from the ith digit of s_y and 0 if they are the same. In other words, we apply $\oplus$ to two states in order to see which agents changed their actions (resulting in a value of 1) and which agents kept (resulting in a value of 0). We will use this vector to count how many agents changed their action between two system states in order to determine the probability of transitioning between these states. Lastly, for each state s_x we compute a

Table 3 Summary of notation used in the Markov chain analysis

Notation	Description	Notation	Description
$\oplus$	Operator on two states; returns a binary vector $\vec{b}_{x,y}$	$\vec{u}^{(t)}$	A probability vector of arriving at each state after t iterations
k	Number of actions	$\vec{u}$	A probability vector of initial states
N	Number of agents in the system	S	Number of system states
$\mathscr{P}$	Transition matrix with elements $\pi_{x,y}$	$a_{i,x}$	Action of agent i in state s_x
$\pi_{x,y}$	Transition probability between states s_x and s_y	s_x	A state in the Markov chain; represents the action of each agent
$\vec{r}_x$	A binary vector of size r; contains 1 for each pair of agents that agree and 0 for pairs that disagree	$\vec{b}_{x,y}$	A binary vector of size N; contains 1 for each agent who has a different action between states s_x and s_y and 0 for agents with the same action between those states
r_{agree}	Number of pairs of agents that have the same action	r	Number of pairs of agents, according to the network topology
r_{disagree}	Number of pairs of agents whose actions differ	α	Probability to select the action of the interacting partner in the next interaction if the actions of the two agents differ

vector $\vec{r}_x$ containing a binary value for each pair of neighboring agents. With a slight abuse of notation, we write that each element $\vec{r}_x[ij]$ of $\vec{r}_x$ indicates whether the corresponding pair of agents i and j have the same action (i.e., they "agree", resulting in a value of 1), or their actions differ (or "disagree", resulting in a value of 0) in state s_x. This vector is necessary in order to determine the elements in our transition matrix $\mathscr{P}$. The vector is computed based on the network topology and the vector's size r is equal to the number of edges in the graph. For example, in a ring topology, we have $r = N$ values, as there are exactly N pairs of neighboring agents (or edges in the graph). In the fully connected topology, on the other hand, there are $r = \binom{N}{2}$ pairs of agents. In this way we can calculate r_{agree}, the number of pairs that agree:

$$r_{\text{agree}} = \sum_{\iota=1}^{r} \vec{r}_x[\iota]$$

which is simply the number of 1s in $\vec{r}_x$. Similarly the number of pairs that disagree is $r_{\text{disagree}} = r - r_{\text{agree}}$. Table 3 summarizes the components of our Markov chain model.

We can now compute the transition probability $\pi_{x,y}$ between each pair of states s_x and s_y in the following manner:

$$\pi_{x,y} = \begin{cases} \dfrac{r_{\text{disagree}}(1-\alpha)^2 + r_{\text{agree}}}{r} & \text{if } \sum_{\iota=1}^{N} \vec{b}_{x,y}[\iota] = 0 \\ \dfrac{\sum_{j=1, j\neq i}^{N} f(a_{j,x}, a_{i,y}) \cdot \alpha(1-\alpha)}{r} & \text{if } \sum_{\iota=1}^{N} \vec{b}_{x,y}[\iota] = 1 \\ \dfrac{\alpha^2}{r} & \text{if } \sum_{\iota=1}^{N} \vec{b}_{x,y}[\iota] = 2 \text{ AND } \vec{r}_{x,y}[ij] = 1 \text{ AND} \\ & \quad \text{AND } a_{j,x} = a_{i,y} \text{ AND } a_{j,y} = a_{i,x} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $f(a_{j,x}, a_{i,y})$ is a function returning 1 if agents i and j are neighbors and $a_{j,x} = a_{i,y}$, and 0 otherwise. Using Eq. 2 we calculate each entry in the transitioning matrix $\mathscr{P}$.

Informally, we distinguish between four different cases, based on the number of agents that changed their action between states s_x and s_y. Equation 2 can be intuitively described in the following manner (in the same vertical order of the 4 cases above):

1. If all agents have the same action between two states (i.e. $\sum_{\iota=1}^{N} \vec{b}_{x,y}[\iota] = 0$) it means that the selected agents have kept their actions depending on whether they agreed or disagreed. We compute the probability of either having the two selected neighbors disagree in s_x and both of them keep their actions in s_y, or the two selected neighbors agree in s_x (in which case they both keep in s_y).
2. If exactly one agent changed its action between two states: we find the probability of exactly that agent i being paired with a neighboring agent j that has a different action, and that the latter action is precisely the action that agent i has shifted to in state s_y while agent j has kept its action.
3. If exactly two neighbors swapped their actions between two states, the probability of that happening is simply $\alpha \cdot \alpha / r$, since both selected agents change their actions, and those two agents are one out of all r possible pairs to be selected.
4. Lastly, the transition probability is 0 between states where the actions of more than two agents differ, as this is not possible, since at every iteration exactly one pair of agents is selected to interact. In addition, $\pi_{x,y}$ is 0 if the actions of exactly two agents differ between states s_x and s_y, if those agents are either not neighbors, or their actions in s_x are not swapped in s_y, but contain other actions instead. Similarly, $\pi_{x,y}$ is 0 if exactly one agent changed its action in s_y to an action that did not exist in s_x.

Note that this Markov chain analysis is generic and can be used to analyze the behavior of arbitrary number of agents and actions in any topology, since $\vec{r}_x$ is the only topology-dependent component (together with r, r_{agree} and r_{disagree}). Thus, for a given topology and number of agents N, one can compute $\vec{r}_x$ and use it in Eq. 2 together with the number of actions k and shift probability α to obtain the elements of the transition matrix $\mathscr{P}$. We will show later how using that matrix we can determine the expected convergence time of the system.

If we assume that all initial states are equally likely, we can set all elements of $\vec{u}$, the vector of initial probabilities, to $1/k^N$, since there are k^N possible states. We can compute the probability of arriving at any individual absorbing state after t iterations using Equation 1. The probability $\Pi^{\text{conv},(t)}$ that a network with N agents and k actions will converge in t iterations is:

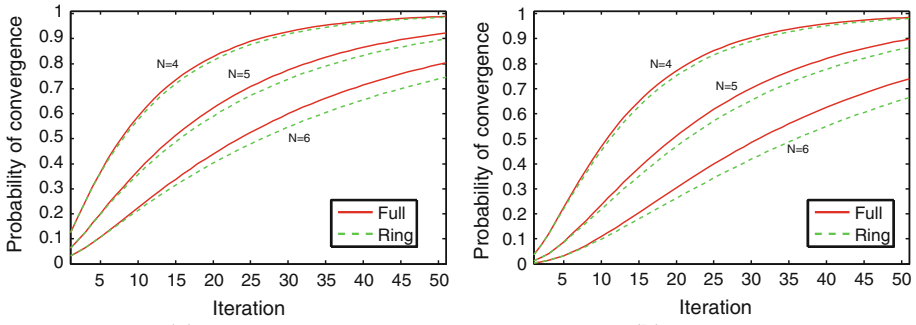
$$\Pi^{\text{conv},(t)} = \sum_{\iota \in \hat{S}} \vec{u}^{(t)}[\iota] \tag{3}$$

where $\hat{S} \subseteq S$ is the set of all goal states in S, and $\vec{u}^{(t)}[\iota]$ is the ιth element of $\vec{u}^{(t)}$. Here we simply sum the probabilities of arriving at all k goal states at iteration t.

Lastly, we arrive at our main analytical result, expressed in the following theorem.

Theorem 2 *The Markov chain of WSLpS is absorbing and agents using WSLpS have a non-zero probability to reach convention in a finite number of iterations.*

Proof Let $A = \{1, 2, \ldots, N\}$ be a finite set of N agents and $K = \{1, 2, \ldots, k\}$ be a finite set of k actions, where a_i^t represents the action of agent i at iteration t. Further, let $C_m^t = \{i | i \in A, a_i^t = m\}$ be the set of all agents i whose action a_i^t at iteration t is $m \in K$. Thus,



(a) $k = 2$ actions. **(b)** $k = 3$ actions.

Fig. 1 Probabilities for $N = 4, 5, 6$ agents with $k = 2, 3$ actions to converge within the first $t = 1, \ldots, 50$ iterations in fully connected and ring topologies for $\alpha = 0.5$

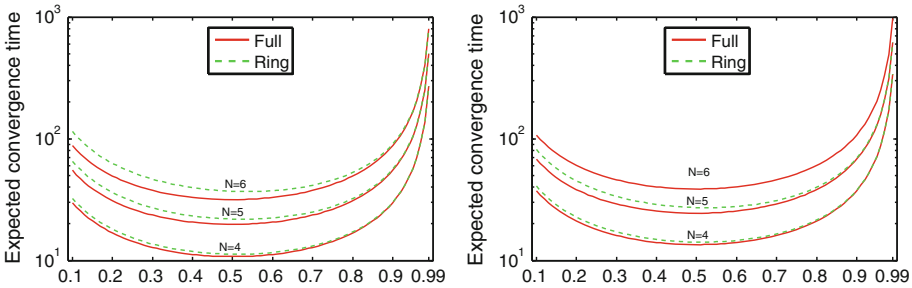
the sets C_m^t are a partitioning of the set of agents A. A goal state (or a state of convention) in our Markov chain is a state where $C_m^t = A$ for a given action $m \in K$ and $C_n^t = \emptyset \, \forall n \neq m$. Since in each goal state there are no conflicts between agents, the probability that any agent i will select a different action in the next iteration is 0. Therefore all goal states are absorbing. In any other non-goal state there is a conflict between at least two neighboring agents. Since $\alpha > 0$ there is a non-zero probability that the system will transition to a different state and therefore these are called transient states. Thus, there are no absorbing states other than the goal states.

A transient state at iteration t implies that there is a conflict between at least two neighboring agents. Therefore, $\exists i, j \in A$ such that i and j are neighbors in the graph and have different actions, i.e. $i \in C_m^t$ and $j \in C_n^t$ for $m, n \in K, m \neq n$. Thus, at iteration $t + 1$ there is a non-zero probability that $C_m^{t+1} = \{j\} \cup C_m^t$, i.e. a neighbor j of agent i will change its action to agree with agent i. Similarly, at iteration $t + 2$ the probability that a neighbor l of agent j will select j's action is larger than 0. In a connected network with N agents there is a non-zero probability that at iteration $t + N$ all neighbors of i, all neighbors of j and so on will select action a_m^{t+N} resulting in $C_m^{t+N} = A$. Thus, an absorbing state can be reached from any transient state (not necessarily in one step) and therefore there are no absorbing loops. Since the system has a finite population N, agents using the WSLpS approach are able to converge in finite number of iterations. □

The above theorem says that our Markov chain is absorbing and that for $t < \infty$, $\Pi^{\text{conv},(t)} > 0$, i.e. there is a non-zero probability that agents will reach convention in a finite number of iterations. Following from the properties of absorbing Markov chains [20] we have that as $t \to \infty$, $\Pi^{\text{conv},(t)} \to 1$.

Note that the absorption probability $\Pi^{\text{conv},(t)}$ can be computed for an arbitrary number of agents and actions and for any network topology, based on the shift parameter α. We show in Fig. 1 the probability of convergence for different number of agents and actions in fully connected and in ring topologies. We see that the fully connected topology has a slightly higher probability of convergence, as compared to the ring topology. We also see that the higher the number of agents or available actions, the lower the probability of convergence. These observations will be confirmed and further analyzed through our simulation studies in larger networks in Sect. 6.1.

Besides the probability of convergence, one can also compute the expected convergence time of agents, i.e. the average number of iterations necessary until all agents select the same



(a) $k = 2$ actions. **(b)** $k = 3$ actions.

Fig. 2 Expected convergence time of agents in fully connected and in ring topologies for different values of α

action. Let $\mathscr{Q}$ be the matrix generated by taking $\mathscr{P}$ and removing all rows and columns that contain a probability of 1. In other words we remove all k rows that contain the probabilities of going from an absorbing state to any state and all k columns with probabilities of reaching an absorbing state from any state. In this way $\mathscr{Q}$ contains the probabilities of transitioning between any pair of transient states in a single step. We then compute the fundamental matrix $\mathscr{N}$ to obtain the expected number of times the process is in each transient state:

$$\mathscr{N} = (\mathscr{I} - \mathscr{Q})^{-1} \tag{4}$$

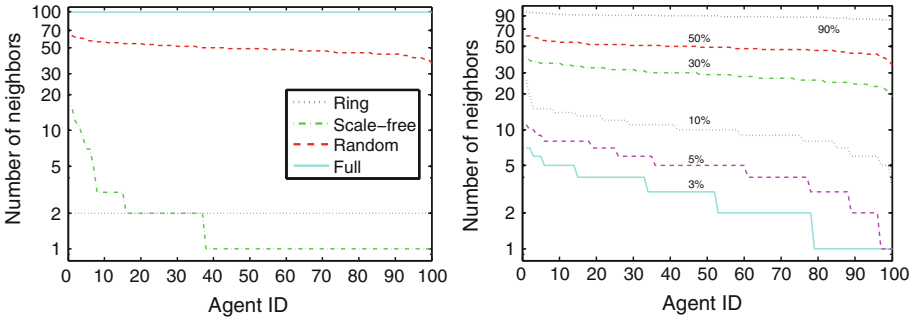
where $\mathscr{I}$ is the identity matrix. Using the fundamental matrix $\mathscr{N}$ we can compute the row vector $\vec{e}$ that gives us for each starting state the expected number of iterations until the chain is absorbed:

$$\vec{e} = \mathscr{N}\vec{1} \tag{5}$$

where $\vec{1}$ is a column vector of all ones. Thus the element $\vec{e}[x]$ shows us the number of iterations before the chain is absorbed when starting in state s_x. Finally, we compute the expected convergence time E based on the initial distribution of states:

$$E = \vec{u}'\vec{e} \tag{6}$$

where $\vec{u}'$ is the transposed vector representing the distribution of starting states. In this way the Markov chain allows us to study the behavior of agents using WSLpS, as well as the effect of the parameter α on the convergence time of the network. We show in Fig. 2 the expected time of agents to reach convention both in the fully connected and in the ring topologies for different values of α. As we predicted in Sect. 5, the value of α that results in the fastest convergence in symmetric asynchronous pairwise interactions is indeed 0.5, since both agents independently update their action. One can notice that for 2-action games the WSLR strategy coincides with WSLpS where $\alpha = 0.5$ under this interaction model. However, we will see later that for $k > 2$, the two approaches differ and WSLpS outperforms WSLR. We will see in Sect. 6.1 that all of the above results will be confirmed in our simulation studies in large networks and that the best value for α is different when using other interaction models that are typical for some engineering applications.



(a) Number of neighbors per agent in different topologies.

(b) Number of neighbors per agent in the random topology at different neighborhood sizes as percent of the total population.

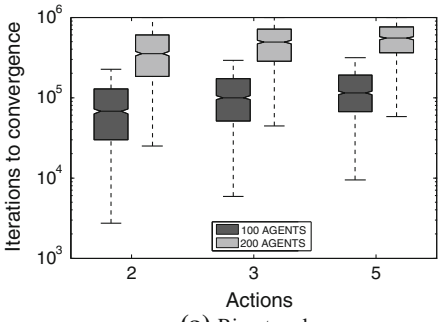
Fig. 3 Distribution of neighbors per agent in networks of 100 agents. Agents are sorted according to the number of neighbors

6 Experiments

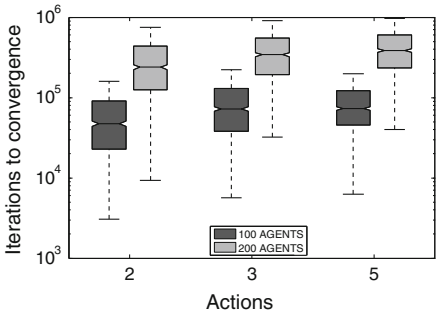
In the following sections we study the effect of various system parameters on the convergence time of our WSLpS approach in different settings. We first investigate symmetric pairwise asynchronous interactions so that we can compare our WSLpS approach to strategies proposed for human populations. We then explore other interaction models, typical for engineering applications and address some common assumptions.

6.1 Experimental setting

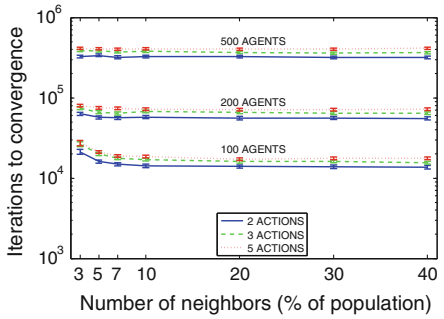
We study four different **topology** types: ring, scale-free, random and fully connected networks. We explore here static topologies, while networks of dynamic connectivity will be addressed in Sect. 6.5. The *ring* topology resembles a common scenario in computer networks where each node has exactly 2 neighbors. It poses an interesting challenge for convention emergence, due to the sparse connectivity of agents and the high network diameter (the longest shortest path between any two agents). Next, we explore *scale-free* networks, which resemble the connectivity between computer agents that represent humans in a social network. Our scale-free networks are generated using the preferential attachment algorithm of Barabasi et al. [3], where each new node is added with exactly one connection to the existing network nodes, based on the degree of the nodes. The number of neighbors per node in the resulting network follows a power-law distribution. We study also the behavior of agents in *random networks* of different densities, in order to understand how densities affect convergence time. Such topologies represent the connectivity between autonomous agents, scattered in an environment, e.g. wireless sensor networks. Lastly, we measure the convergence process in *fully connected* networks, as it is often done in the literature on coordination games. This extreme case resembles the interconnectivity in some engineering applications where everybody can interact with everybody. The number of neighbors for each agent in the above four topologies is displayed in Fig. 3. Note that for clarity in Fig. 3a we show only particular instances for the scale-free and random networks, since these networks are generated by stochastic processes. The same holds for Fig. 3b where we show different random topologies. In our experiments, however, we generate a different network in each sample run.



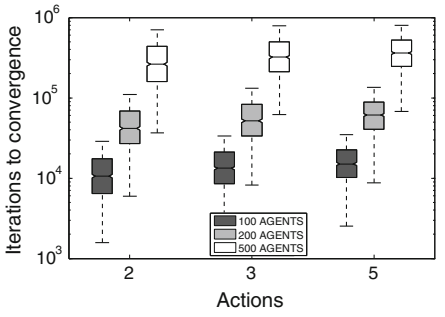
(a) Ring topology.



(b) Scale-free topology.



(c) Random topology with different densities. Error bars show the 95% confidence interval of the mean.



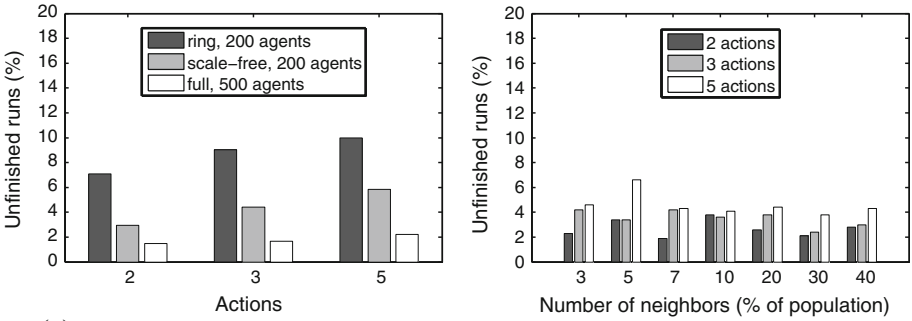
(d) Fully connected topology.

Fig. 4 Convergence time for different topologies under asynchronous pairwise interactions. On each box plot, the central mark is the median, the edges of the box are the q_1 and q_3, i.e. the 25th and 75th percentiles. The notches show the 95 % confidence interval of the median. The lower and upper whiskers extend to the most extreme data points not considered outliers, i.e. to the data points adjacent to $q_1 - (q_3 - q_1)$ and $q_3 + (q_3 - q_1)$, respectively. Outliers are not shown

In the above topologies we vary the **number of agents** and **number of actions** available to those agents in order to investigate the scalability of our approach. We explore networks of 100, 200 and 500 players where agents can have 2, 3 or 5 actions. The convergence times for each parameter configuration are averaged over 1,000 runs of Algorithm 1 in MATLAB. 1,000 runs with a given parameter configuration we call a sample. Each run ends either when a convention emerges, or when a maximum of 10^6 iterations is reached ($T_{max} = 10^6$), in which case the run is not counted towards the mean of the sample. The sample is considered only if at least 90 % of the runs have finished within T_{max} iterations. The action for each agent is initialized uniformly random from the available actions. As stated above, the performance measure of the system is the number of iterations until the actions of all agents converge. We set here the parameter α, i.e. the shift probability upon conflict, to 0.5, as this value gives the shortest convergence time in symmetric pairwise interactions (cf. Fig. 2).

6.2 Results

Figure 4 shows the convergence duration of agents arranged in different topologies under the pairwise asynchronous interaction model. This is also the classic experimental setting reported in literature (see Sect. 2). In all topologies with 100 agents all runs converged within



(a) Unfinished runs in ring, scale-free and fully connected topologies.

(b) Unfinished runs in random topologies with 500 agents at different densities.

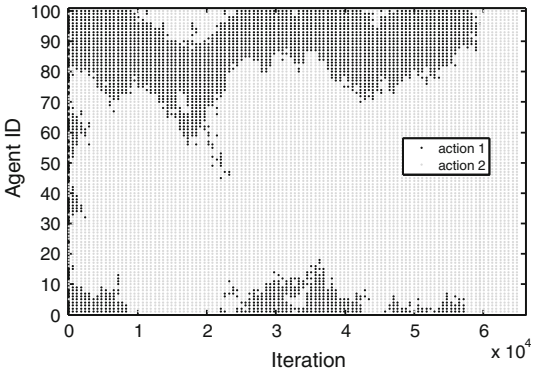
Fig. 5 Percent of runs of WSLpS that have not finished within $T_{max} = 10^6$ iterations from the experiments presented in Fig. 4

the imposed limit of $T_{max} = 10^6$ iterations. In addition, all runs with 200 agents converged in the fully connected and random topologies. In all other reported settings in Fig. 4, i.e. ring and scale-free with 200 agents and random and fully connected with 500 agents, at least 90 % of the runs converged within T_{max} iterations. In Fig. 5 we display the percentage of runs that have not converged within the specified limit. Nevertheless, as stated in Theorem 2, convergence *is* guaranteed in finite number of iterations.

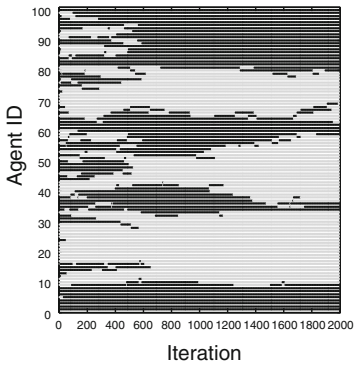
We observe here the scalability of our approach with respect to the number of agents and actions—convergence time increases linearly with the number of actions, but it is exponential in the population size (Note that vertical axes are in the log scale). Another observation is that densely connected networks (Fig. 4c, d) converge on average faster than networks with sparse connectivity (Fig. 4a, b). Similar observations are reported by other researchers applying different learning algorithms [9,41]. This phenomenon is particularly visible in Fig. 4c where random networks with few neighbors per agent converge slower than networks where agents have more neighbors. The reason for this behavior is that agents in denser networks behave in response to the actions of larger groups of neighboring agents and therefore have better chance to arrive at a mutually beneficial outcome. In contrast, an agent with only few neighbors, for example, receives feedback based on the actions of agents in a very small portion of the network. Those limited interactions result in less conflicts, as the locally most common action varies only little, and therefore actions of agents in small neighborhoods are reinforced, leading to slower convergence of the whole network. However, the effect of this phenomenon saturates relatively quickly, i.e. the convergence time does not decrease further for networks where neighborhoods are larger than 10 % of the population. In denser networks the faster propagation of conventions is counterbalanced by the increased probability of conflict, due to the large neighborhoods.

To have a better understanding of the convergence process at a finer scale, we present in Fig. 6 the behavior of agents in a typical simulation run of Algorithm 1. Figure 6 displays the action of agents during learning in the ring topology with 100 agents and 2 available actions. Figure 4a indicates that the mean convergence time in the above configuration is between 6×10^4 and 7×10^4 iterations, which is what we observe in our simulation run in Fig. 6. Note that despite their large number, interactions typically happen quite fast in computer systems. For example, wireless devices can exchange messages in the order of 100s per second.

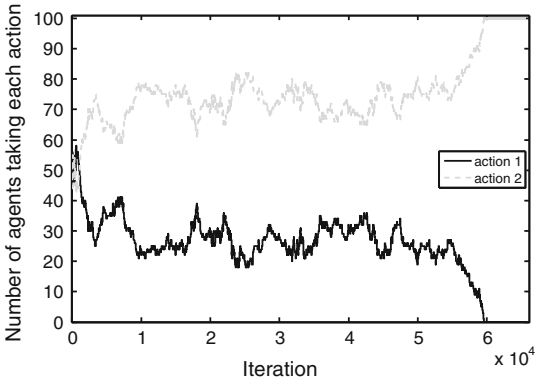
Each value on the vertical axis of Fig. 6a represents a different agent. In other words, *Agent ID* is not a continuous variable, but simply shows the individual agents. Each dot in the latter



(a) The action of each agent throughout the simulation process.



(b) A detail of the first 2000 iterations.

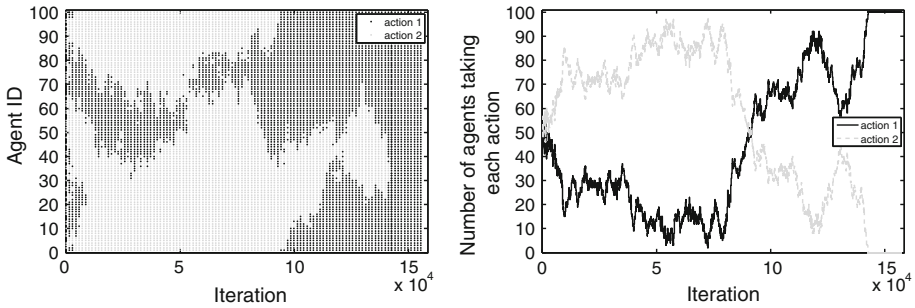


(c) The number of agents taking each action at each iteration.

Fig. 6 Results from a single (typical) simulation run of Algorithm 1 in the ring topology with 100 agents and 2 available actions. Pairwise asynchronous interaction model with $\alpha = 0.5$

figure stands for the action of each agent at the corresponding iteration. A black dot represents action 1 and a gray dot—action 2. As mentioned in Sect. 4.2, these actions are initialized at random. In Fig. 6b we show a detailed view of the first 2,000 iterations. It can be observed that at iteration 1 agents are randomly assigned actions 1 and 2. Since agents are arranged in a ring topology, consecutive agent IDs in Fig. 6a are also neighbors in the network. An interesting observation is that shortly after the start, large contiguous clusters of neighboring agents tend to select the same action. Only agents on the border between different sections experience conflicts and therefore change actions. This behavior demonstrates the essence of our WSLpS approach—agents with "successful" actions will keep selecting the same action, while those who experience conflicts have a non-zero probability of changing.

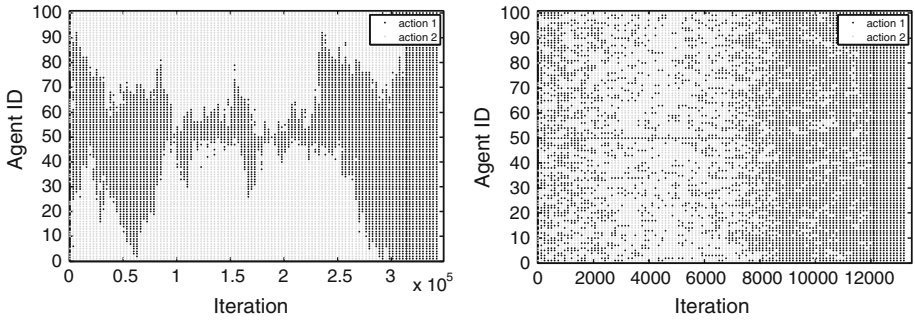
While Fig. 6a shows the action of each agent through time, Fig. 6c reports the number of agents taking each action at every iteration during the same simulation run. One can observe that at iteration 1 equal number of agents select each action, as stated earlier. We notice that very quickly action 2 becomes dominant and later on all agents learn to select it. We see also large fluctuations in the number of agents who select the same action. The reason for these fluctuations is the probabilistic component of our WSLpS. We see that around iteration 6×10^4 all agents learn to select action 2 and thus a convention has emerged. We ran the



(a) The action of each agent throughout the simulation process.

(b) The number of agents taking each action at each iteration.

Fig. 7 An example in ring topology showing that a convention can be reversed unless all agents adopt it



(a) Ring topology.

(b) Fully connected topology.

Fig. 8 Two examples illustrating that the theorem of impossibility of distributed consensus with one faulty process [14] does not hold when using WSLpS. In both networks agent 50 is stuck with action 1 and both systems converge

simulation for 10^4 steps more in order to illustrate that agents continue to select the same action and therefore do not escape the convention, even though individual agents are not aware of the global system outcome. Maintaining conventions stable is an important feature that most strategies proposed in the literature on coordination games are lacking.

We present here another sample run that demonstrates a problem with the common assumption in literature that conventions are reached when at least 90 % of the agents select the same action. As we argued in Sect. 2, unless 100 % of the agents select the same action, some agents still experience conflicts. If agents apply stochastic strategies, such as HCR (used by [36]), ε-greedy (used by [42]), or WSLpS, conventions can still be "reversed", as it is evident in Figs. 7 and 8. We notice that using WSLpS although in the first 9×10^4 iterations action 2 is dominant and is selected by more than 90 % of the agents, later on the majority of the agents learn to select action 1 instead. Other coordination algorithms in literature exhibit similar properties. Therefore, the convergence time should be considered as the period from an initial random state until all agents reach convention.

As computer systems cannot be assumed to be fault-free, we study here convention emergence in the presence of a fault, which may occur e.g. due to physical or electronic damage. We see in Fig. 8 that the multi-agent system can still converge even if one agent is stuck with the same action and is unable to change it. In Fig. 8a we show the process of conver-

Table 4 Comparison between the performance of HCR, Q-learning and WSLpS in different topologies with 100 agents and 2 actions

Topology	HCR	Q-learning	WSLpS
Ring	Does not converge	Not reported	$\mathbf{6.5 \times 10^4}$
Scale-free	Does not converge	5×10^6	$\mathbf{5 \times 10^4}$
Random, 10 %	Does not converge	2.2×10^8	$\mathbf{1.5 \times 10^4}$
Fully connected	$\mathbf{8 \times 10^2}$	2×10^5	10^4

Agents are involved in symmetric asynchronous pairwise interactions. The fastest convergence time in each topology is shown in bold

gence in a ring topology, where one agent always selects the same action, regardless of the play of others. Similarly, in Fig. 8b one agent in the fully connected network is stuck with the same action. Note that we focus here on systems owned by one user and therefore any misbehaving agents are considered faulty, rather than malicious. Franks et al. [15], on the other hand, consider systems of heterogeneous ownership where the authors investigate how a small group of malicious agents can manipulate convention emergence. Similarly, we see how our faulty agents in Fig. 8 in fact unintentionally steer the convention to the action they are stuck with. Contrary to the theorem of Fischer et al. [14], in both of the above settings the system converges to a convention, despite the presence of a fault. Of course if multiple agents are stuck each with a different action, there will be no absorbing states and therefore the system will never converge, unless the faulty agents can be excluded from interactions. Nevertheless, more detailed studies are necessary in order to investigate the fault tolerance of WSLpS.

6.3 Comparison

A number of approaches have been proposed for decentralized coordination (see Table 1 in Sect. 2). Here we will focus on two algorithms of particular interest. We compare here with one of the earliest and most commonly used algorithm—HCR, as well as with the most recent one and the only to consider a convergence threshold of 100 %—Q-learning. The results can be seen in Table 4 for the most common convention emergence settings in literature.

Shoham and Tennenholtz [36] introduced the HCR rule that lets agents select the most successful action in the last l iterations, where l is the memory size of agents. The authors prove both the convergence of HCR in fully connected networks, as well as the fact that conventions are absorbing states of the system. Although HCR outperforms WSLpS in fully connected topologies, HCR's convergence is not guaranteed in restrictive topologies, such as ring, random, or scale-free graphs. In these topologies, the HCR rule causes what Villatoro et al. [42] refer to as "the frontier effect", where some neighboring agents cannot resolve their conflicts, as they each support the action of the rest of their neighbors. The latter authors attempt to resolve that problem by modifying the reward signal agents receive from interactions, and by applying the Q-learning update rule in combination with the ε-Greedy actions selection policy.

Although the ε-Greedy policy resolves the above conflicts, conventions are no longer absorbing states of the system. In addition, despite our best efforts and personal communication with the first author, we were unable to reproduce the results reported in their article. Nevertheless, our experimental settings are identical[1], and therefore we can rely on the data

[1] Confirmed through personal communication with the first author.

presented in Villatoro et al. [42]. The only difference to our setting is that Villatoro et al. [42] count 1 iteration when all agents have interacted at least once in a round-robin fashion (which we consider to be rather unrealistic in a decentralized setting), while in our studies exactly two random agents interact in 1 iteration. We found no significant difference in convergence time between round-robin and random agent selection when using our approach. Therefore we multiply their results by the number of agents in the network in order to obtain the convergence time expressed in number of pairwise interactions. From Figs. 2, 4 and 9 in the latter article we can read the convergence times of random, fully connected and scale-free topologies, respectively. We see that in all cases WSLpS outperforms their algorithm.

Lastly, in the 2-action case studied above, WSLpS with $\alpha = 0.5$ coincides with WSLR. In general, for k actions WSLR resembles WSLpS where $\alpha = 1/k$. As we stated in Sect. 5.2, fastest convergence is achieved when $\alpha = 0.5$ for any number of actions (cf. Fig. 2), thus WSLpS outperforms WSLR for $k > 2$ actions. Similarly, WSLpS outperforms WSLS, as the latter strategy sets α to 1. We do not explicitly present a comparison table for these algorithms, as WSLS and WSLR are simply different parameter instantiations of WSLpS.

6.4 Other interaction models

So far we assumed that agents interact pairwise and that both agents are aware that the interaction is taking place and thus both agents update their actions. This setting is common in the literature on convention emergence, and is relevant for computer systems that rely on peer-to-peer communication. Communication between wireless devices, however, affects all other wireless devices in range, thus interactions cannot be considered as pairwise, but are **multi-player** instead. This type of one-to-many encounters is rarely studied in literature, but often occurs in practice. Moreover, due to the nature of radio communication, wireless devices cannot always be aware that an interaction is taking place (e.g. due to interference) and therefore interactions are no longer symmetric. For this reason we say that only the "initiator" of an interaction (e.g. the one who sends the wireless signal) updates its action. All other neighbors in range do not update their actions, as we cannot assume that they are aware of the communication, or who initiated it. A similar model of **asymmetric** interactions is used by [7], where each agent plays a 2-player game with all its neighbors and then computes the sum of these bilateral games' payoffs. Similarly, Villatoro et al. [42] experiment with the setting where in pairwise interactions only the "first" agent updates its action, while the "second" agent ignores the obtained payoff from the game.

An example of a coordination game with asymmetric multi-player interactions is a wireless sensor network where nodes coordinate on reserving the same broadcast channel for control messages. In this example, as in Bramoullé et al. [7], the payoff signal to each agent depends on the ratio of neighbors selecting the same action versus all neighbors. Agents can then use the WSLpS strategy to select their actions. The pseudo-code of this strategy is presented in Algorithm 4 for the multi-player interaction model. While in our initial experimental setting exactly two agents are selected to play and both update their actions (see Algorithm 1), here only one agent is selected to play with all its neighbors and only that agent updates its action. Similarly to the pairwise model, agents with high payoff will keep their last action unchanged in the next iteration. The only difference to the algorithm for pairwise interactions is the way agents change their actions upon conflict. While in pairwise interactions (Algorithm 2) the agents probabilistically shift to the action of their neighbor upon "lose", in multi-player interactions the agent probabilistically selects the majority action in its neighborhood, breaking ties randomly, and with a small probability ε it selects a different action at random. The latter

behavior is necessary in order to prevent constant oscillations of actions in some topologies. In Mihaylov [27] we study in more detail how ε affects convergence in multi-player interactions.

Algorithm 3 function *updateAction* for the multi-player interaction model

Input: payoff p_i and action a_i of agent i from the latest interaction as well as a vector $\vec{a_j}$ containing the actions of all neighbors j
Output: the new action a_i of agent i

$rnd \leftarrow generateRandomNumber(0, 1)$
$\Pi_i \leftarrow \max(\alpha - p_i, 0)$
if $rnd < \Pi_i$ **then**
    $rnd \leftarrow generateRandomNumber(0, 1)$
    if $rnd < \epsilon$ **then**
        $a_i \leftarrow selectNewRandomAction(a_i)$
    else
        $a_i \leftarrow selectMajorityAction(\vec{a_j})$
    end if
end if
return a_i

Algorithm 4 function *updateAction* for the multi-player interaction model

Input: payoff p_i and action a_i of agent i from the latest interaction as well as a vector $\vec{a_j}$ containing the actions of all neighbors j
Output: the new action a_i of agent i

$rnd \leftarrow generateRandomNumber(0, 1)$
$\Pi_i \leftarrow \max(\alpha - p_i, 0)$
if $rnd < \Pi_i$ **then**
    $rnd \leftarrow generateRandomNumber(0, 1)$
    if $rnd < \varepsilon$ **then**
        $a_i \leftarrow selectNewRandomAction(a_i)$
    else
        $a_i \leftarrow selectMajorityAction(\vec{a_j})$
    end if
end if
return a_i

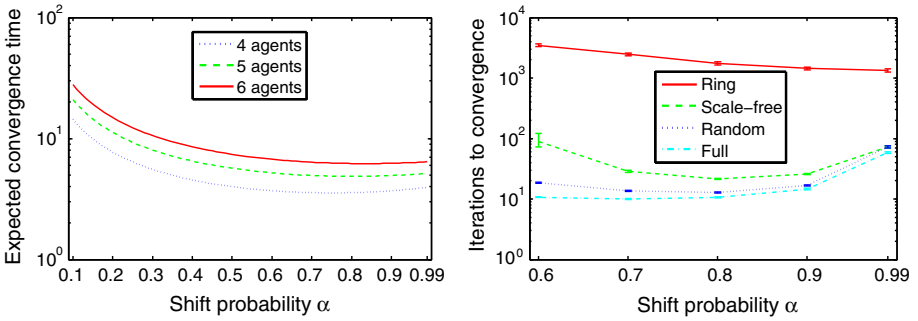
As only one agent updates its action at a given iteration, the best chance of convergence is when we set α to 1. Unlike symmetric interactions, where the best value for α is 0.5 to resolve symmetry, here the agent should always switch to the majority action in its neighborhood and with a small probability select an action at random upon conflict. Our experimental studies (not displayed here) confirm that $\alpha = 1$ results in the fastest convergence time in all topologies. Therefore, in this setting one can apply the WSLS strategy, as it coincides with WSLpS where $\alpha = 1$.

Next we address the assumption of **common observability**, which is also used by other researchers [9,35,41]. In pairwise interactions we assumed that each agent is able to observe its neighbor's action and probabilistically switch to that action upon conflict. Similarly, with multi-player asynchronous interactions the initiator may observe and switch to the majority action in its neighborhood. These assumptions are relevant in systems where such knowledge comes "for free", e.g. virtual fireflies synchronizing their flashes [22]. However, in many domains observation is costly and needs to be used sparingly [10]. In others, observation is

Algorithm 5 function *updateAction* for the multi-player interaction model, no observation

Input: payoff p_i and action a_i of agent i from the latest interaction
Output: the new action a_i of agent i

1: $rnd \leftarrow generateRandomNumber(0, 1)$
2: $\Pi_i \leftarrow \max(\alpha - p_i, 0)$
3: **if** $rnd < \Pi_i$ **then**
4: $\quad a_i \leftarrow selectNewRandomAction(a_i)$
5: **end if**
6: **return** a_i



(a) Expected convergence time of agents in a fully connected topology. Calculated using Markov chain analysis.

(b) Empirical convergence time of 100 agents in different topologies. *Error bars* show the 95% confidence interval of the mean.

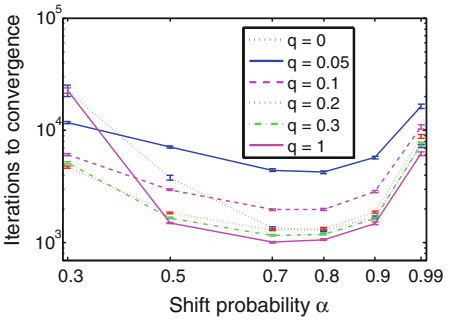
Fig. 9 Multi-player synchronous asymmetric interaction model with 2 actions per agent for different values of α

Table 5 Best values of the shift probability α in WSLpS for different interaction models

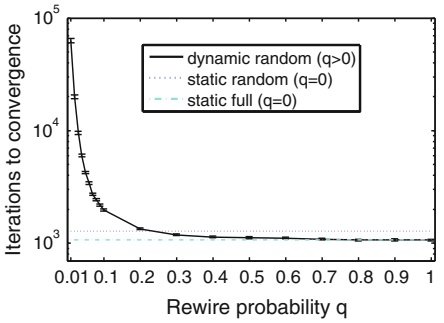
		Pairwise	Multi-player
Synchronous	Asymmetric	0.7–0.9	0.7–0.9
Asynchronous	Symmetric	0.5	N/A
	Asymmetric	1	1

even impossible, e.g. a wireless node cannot determine the channel on which its neighbor is listening for messages, or whether a neighbor is listening at all, as opposed to sleeping. The coordination algorithm, therefore, needs to consider these limitations and still enable the system to reach convergence. Nevertheless, conflicts, such as transmitting to an agent who is sleeping, can still be detected. From here onwards we relax the assumption of common observability and show how agents using our WSLpS approach can all converge to the same action. We show in Algorithm 5 the pseudo-code of this approach for the multi-player interaction model. Upon conflict, the agent will select a different action at random with probability α, rather than select the most common action, as it is not aware of the latter. The convergence guarantees and the absorbing state property still hold, as we prove in Mihaylov [27] using similar Markov chain analysis to that in Sect. 5.2.

So far we assumed that time is continuous and that at any given time instant $\triangle t$ at most one agent (or one pair of agents) updates its action, due to the asynchronous nature of interactions. Some wireless communication protocols, however, require that actions are performed simultaneously using synchronized internal clocks of agents [26,43]. Maintaining synchronized clocks in a decentralized system is possible with a relatively low communication overhead



(a) Empirical convergence time for different values of α and q.



(b) Detailed results for different values of q when $\alpha = 0.8$.

Fig. 10 Multi-player synchronous asymmetric interactions with 100 agents and 2 actions per agent in dynamic random topologies. *Error bars* show the 95 % confidence interval of the mean

[24]. Using protocols such as Time Division Multiple Access, in one iteration all agents interact and update their actions **synchronously**, although computation is still asynchronous. In these settings agents can still apply Algorithm 5 using an appropriate shift probability.

Figure 9 shows how α affects the convergence time in different topologies under this multi-player synchronous asymmetric interaction model. Note that here in one iteration all agents interact, while in the asynchronous model only one agent (or a pair) updates its action in any given iteration. We see that networks larger than 2 agents converge faster with $0.7 < \alpha < 0.9$ (cf. Fig. 9a), due to the synchronous updates of agents. Our empirical results in large topologies confirm this trend (cf. Fig. 9b) where we see that the fastest convergence time is achieved with α between 0.7 and 0.9, rather than 0.5 as in symmetric interactions, or 1.0 as in asynchronous interactions. Therefore, neither WSLR nor WSLS can outperform WSLpS in these settings. Table 5 summarizes the values of the shift probability α that result in the fastest convergence in each interaction model.

6.5 Dynamic topology

Until now we studied coordination games in static topologies, which are suitable to model interactions between stationary agents, e.g. nodes in a wireless sensor network. In this section we explore the convergence time of dynamic topologies, i.e. when agents **dynamically rewire** their connections [32]. This scenario resembles interactions between mobile agents, such as those in swarm robotics. At every iteration, each edge (or link) in the graph has a probability $q \in [0, 1]$ to rewire, i.e. to change one of its two vertices to a uniformly random other vertex in the graph. This probability is constant and is the same for all links. In the limits, when $q = 0$ the topology remains static, while with $q = 1$ each link always rewires at every iteration. Note that dynamic graphs are no longer guaranteed to be connected. Nevertheless, we still require that all (disconnected) groups of agents converge to the same action. This is still possible, as the network constantly rewires and agents encounter actions of others.

We display in Fig. 10 the convergence time of agents in a random topology with initial average neighborhood size of 10 % for different values of the rewire probability q. We see that small positive values of q yield convergence times much slower than those in a static topology (i.e. when $q = 0$). For larger values of q, on the other hand, the dynamic network converges faster than the static one. The reason for this result is the following. Small rewire probabilities have a negative effect on the spreading of conventions, as converged agents are

occasionally disturbed by rewiring them to agents with different actions. Larger values of q, however, result in more frequent topology changes and thus help spreading conventions faster to all parts of the network. In fact, when $q \to 1$ the dynamic network behaves much like a fully connected one, since every couple of iterations on average each agent has the chance to meet all other agents in the system. We see in Fig. 10b that in our experimental setting when $q > 0.6$ the convergence time of the dynamic network is comparable to that of a static fully connected one. Lastly, the value of q, for which dynamic random topologies require relatively the same amount of iterations to converge as static random topologies, depends the network size and its density. In our scenario, when $q \approx 0.25$ the convergence time of the dynamic network is comparable to that of the static one.

6.6 Discussion

We saw in this article that the assumptions in the literature on coordination games do not necessarily hold for coordination problems in engineering applications. The application domain imposes certain restrictions on the knowledge available to agents and on the way they interact. We saw that our WSLpS approach copes well with such restrictions, but the different interaction models affect the convergence time of agents. The designer of such decentralized systems, therefore, needs to be aware of the restrictions imposed by the real-world application domain and take them into account when designing the coordination mechanism.

We studied here agent interactions in both static and dynamic networks. In static networks we observed that convergence time has inverse relationship with network density—the higher the connectivity of the network, the less iterations necessary for convergence. Similar effect is present in dynamic topologies where higher mobility, i.e. higher rewire probability, leads to faster convergence. Therefore, if the system designer has control over the network topology, our results suggest that he/she should prefer denser static networks or highly mobile dynamic ones.

Since we are focusing on applications where all agents are owned by the same user, we assume there are no malicious agents. The theorem regarding the impossibility of distributed consensus with one faulty process [14] holds here only if it can be assumed that the fault causes the agent to always select its action at random. Unless the latter behavior can be detected, hardly any algorithm can cope with such a fault. However, as we saw in Fig. 8, the theorem does not hold if an agent is faulty by being "stuck" with the same action, regardless of the play of others. The same holds if multiple agents are stuck all with the same action. An analogy to a faulty agent, used in the literature on opinion dynamics, is that of a "zealot"—an individual that does not change its opinion [29]. Using our approach, the system still has non-zero probability to converge to the state where all agents select the action of the faulty/zealot players. Faults due to delayed communication (e.g. an agent acting based on old observations) may affect the convergence time of agents, but not necessarily prevent the system to reach convention, as agents probabilistically shift upon conflict. Lastly, agents that are dead, e.g. due to depleted battery, or with a damaged communication device, simply cannot participate in interactions with others and therefore such faulty agents will not disturb the system.

WoLF [6] is the principle of applying a variable learning rate to different learning approaches, in order to improve their convergence properties. Sen and Airiau [35], for example, apply WoLF to Policy Hill Climbing and compare it to other learning techniques. WoLF cannot be applied to WSLpS, as we use no notion of learning rate and have a different notion of a "winning" action. A WoLF agent defines winning actions as those, whose *expected* payoff is larger than the equilibrium payoff. In addition, according to the underlying action

selection policy (e.g. ε-Greedy), an agent may still change its action when winning, possibly slowing down the emergence of conventions in restricted topologies.

Lastly, we assumed that all feedback is binary and convention states are equally desirable (e.g. driving on the left vs. driving on the right), as long as everyone does the same. Nevertheless, coordination problems may exist, in which some conventions are more preferred by all agents than other conventions. One example is coordinating on the wireless channel to use for control messages in wireless sensor networks, where some channels may have better quality than others. In those settings agents would still have a higher probability to select the best channel, given that the payoff is scaled in the range [0,1]. Once converged, however, agents will not re-learn to find a better channel, if one is introduced after convergence, as long as the original convention state is not disturbed.

7 Conclusions

The literature on convention emergence is mainly inspired by coordination games observed in human populations where agents are typically involved in asynchronous symmetric pairwise interactions. We saw in this article that interactions between computer agents can take many different forms, depending on the application domain, calling for a more detailed study of coordination games in the context of engineering applications. Coordination games played between computer agents have different assumptions on the information available to agents, which in turn influences convergence time. Thus, coordination algorithms for engineering applications need to be (1) flexible enough in order to deal with different coordination challenges induced by the problem domain, (2) powerful enough, in order to cope with limited information, (3) and simple enough in order to be implemented by highly constrained agents.

Our main objectives in this article were to propose a decentralized approach for fast on-line convention emergence in multi-agent systems, to analyze its convergence properties and to evaluate the behavior of agents through an extensive simulation study. Our approach falls into the prescriptive category of the coordination game literature and is called WSLpS, generalizing two well-known strategies in game theory—WSLS and WSLR. The probabilistic component of our approach, however, allows for a whole spectrum of strategies, two of which are WSLS and WSLR.

Our empirical results suggest that for certain values of this probabilistic component WSLpS yields strategies that outperform both WSLS and WSLR, as well as some state-of-the-art coordination approaches. We showed that using WSLpS, within only a short number of iterations, and under local interactions, agents involved in a repeated pure coordination game are able to reach a mutually beneficial outcome on-line without a central mediator. Using the theory of Markov chains we proved that our WSLpS approach has a non-zero probability to converge to a pure coordination outcome in finite number of iterations and that it always converges in the limit. With our experiments in both static and dynamic topologies we demonstrated that WSLpS performs well in different settings under limited knowledge and imposes minimal system requirements, due to its simplicity. Another desirable property of our approach is that conventions become absorbing states of the system, so that once all agents learn to select the same action, they will no longer change actions and escape the convention. Nevertheless, if the convention is somehow externally disrupted, agents are still guaranteed to converge to a (possibly different) convention.

While initial experiments in our previous work have shown the fault tolerance of WSLpS in actual engineering scenarios, such as wireless sensor networks, the robustness of our approach needs to be further investigated in more detail in a wider array of engineering applications,

e.g. in mobile and peer-to-peer networks. Another interesting research perspective is a study on the evolution and dynamics of coordination in the context of private information and self-interest. For example, in the domain of smart grids our assumption of altruistic agents does not hold, as agents pursue their own self interest, thereby influencing the dynamics of the whole system. One needs to further explore to what extend our convergence proofs still hold in the presence of self-interested agents.

References

1. Axelrod, R. (1984). *The evolution of cooperation*. New York: Basic Books.
2. Axelrod, R. (1986). An evolutionary approach to norms. *The American Political Science Review*, *80*(4), 1095–1111.
3. Barabasi, A. L., Albert, R., & Jeong, H. (1999). Mean-field theory for scale-free random networks. *Physica A: Statistical Mechanics and its Applications*, *272*(1–2), 19.
4. Barrett, J., & Zollman, K. J. S. (2009). The role of forgetting in the evolution and learning of language. *Journal of Experimental & Theoretical Artificial Intelligence*, *21*(4), 293–309.
5. Bendor, J., Mookherjee, D., & Ray, D. (1994). Aspirations, adaptive learning and cooperation in repeated games. Tech. rep., Tilburg University, Center for Economic Research.
6. Bowling, M., & Veloso, M. (2002). Multiagent learning using a variable learning rate. *Artificial Intelligence*, *136*(2), 215–250.
7. Bramoullé, Y., López-Pintado, D., Goyal, S., & Vega-Redondo, F. (2004). Network formation and anti-coordination games. *International Journal of Games Theory*, *33*(1), 1–19.
8. Brooks, L., Iba, W., & Sen, S. (2011). Modeling the emergence and convergence of norms. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence* (Vol. 1, pp. 97–102). Menlo Park, CA: AAAI Press.
9. Castellano, C., Fortunato, S., & Loreto, V. (2009). Statistical physics of social dynamics. *Reviews of modern physics*, *81*(2), 591.
10. De Hauwere, Y.M. (2011). Sparse interactions in multi-agent reinforcement learning (Ph.D. thesis, Vrije Universiteit Brussel, 2011).
11. De Vylder B. (2007). The evolution of conventions in multi-agent systems (Ph.D. thesis, Vrije Universiteit Brussel, 2007).
12. Delgado, J., Pujol, J., & Sanguesa, R. (2003). Emergence of coordination in scale-free networks. *Web Intelligence and Agent Systems*, *1*(2), 131–138.
13. Farinelli, A., Rogers, A., Petcu, A., & Jennings, N. R. (2008). Decentralised coordination of lowpower embedded devices using the max-sum algorithm. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent System* (pp. 639–646). Richland, SC.
14. Fischer, M. J., Na, Lynch, & Paterson, M. S. (1985). Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, *32*(2), 374–382.
15. Franks, H., Griffiths, N., & Jhumka, A. (2013). Manipulating convention emergence using influencer agents. *Journal of Autonomous Agents and Multi-Agent Systems*, *26*(3), 315–353.
16. Grenager, T., Powers, R., & Shoham, Y. (2002). Dispersion games: general definitions and some specific learning results. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence* (pp. 398–403). Alpern: AAAI Press.
17. de Jong, S., Uyttendaele, S., & Tuyls, K. (2008). Learning to reach agreement in a continuous ultimatum game. *Journal of Artificial Intelligence Research (JAIR)*, *33*, 551–574.
18. Karandikar, R., Mookherjee, D., Ray, D., & Vega-Redondo, F. (1998). Evolving aspirations and cooperation. *Journal of Economic Theory*, *80*(2), 292–331.
19. Kelley, H., Thibaut, J., Radloff, R., & Mundy, D. (1962). The development of cooperation in the minimal social situation. *Psychological Monographs: General and Applied*, *76*(19), 1–19.
20. Kemeny, J., & Snell, J. (1969). *Finite Markov chains*. New York: VanNostrand.
21. Kittock, J. (1993). Emergent conventions and the structure of multi-agent systems. In *Proceedings of the 1993 Santa Fe Institute Complex Systems Summer School* (Vol. 5, pp. 1–14). Citeseer.
22. Knoester, D. B., & McKinley, P. K. (2009). Evolving virtual fireflies. In *Proceedings of the 10th European Conference on Artificial Life*, Budapest, Hungary.
23. Kojima, F., & Takahashi, S. (2007). Anti-coordination games and dynamic stability. *International Game Theory Review*, *9*(4), 667–688.

24. Lemmens, B., Steenhaut, K., Ruckebusch, P., Moerman, I., & Nowé, A. (2012). Network-wide synchronization in wireless sensor networks. In *Proceedings of the 19th IEEE Symposium on Communications and Vehicular Technology in the Benelux*.
25. Lewis, D. (1969). *Convention: A philosophical study*. Cambridge: Harvard University Press.
26. Lu, G., Krishnamachari, B., & Raghavendra, C. (2004). An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks. In *Proceedings of the 18th International Symposium on Parallel and Distributed Processing* (p. 224).
27. Mihaylov, M. (2012). Decentralized coordination in multi-agent systems (Ph.D. thesis, Vrije Universiteit Brussel, 2012).
28. Mihaylov, M., Le Borgne, Y. A., Tuyls, K., & Nowé, A. (2013). Reinforcement learning for self-organizing wake-up scheduling in wireless sensor networks. *Agents and artificial intelligence* (Vol. 271, pp. 382–396). Berlin, Heidelberg: Springer.
29. Mobilia, M. (2003). Does a single zealot affect an infinite group of voters? *Physical Review Letters*, *91*(2), 028,701.
30. Mukherjee, P., Sen, S., & Airiau, S. (2008). Norm emergence under constrained interactions in diverse societies. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems* (Vol. 2, pp. 779–786). International Foundation for Autonomous Agents and Multiagent Systems.
31. Nowak, M., & Sigmund, K. (1993). A strategy of Win-Stay, Lose-Shift that outperforms tit-for-tat in the Prisoner's Dilemma game. *Nature*, *364*, 56–58.
32. Savarimuthu, B. T. R., Cranefield, S., Purvis, M. K., & Purvis, M. A. (2009). Norm emergence in agent societies formed by dynamically changing networks. *Web Intelligence and Agent Systems*, *7*(3), 223–232.
33. Schelling, T. C. (1960). *The strategy of conflict*. Cambridge: Harvard University Press.
34. Segbroeck, S.V., Santos, F.C., Lenaerts, T., Pacheco, J.M. (2009) Emergence of cooperation in adaptive social networks with behavioral diversity. In Proceedings of the 10th European Conference on Artificial Life (ECAL) (pp. 434–441).
35. Sen, S., & Airiau, S. (2007). Emergence of norms through social learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence* (pp. 1507–1512).
36. Shoham, Y., & Tennenholtz, M. (1993). Co-learning and the evolution of social activity. Tech. rep. Stanford University.
37. Shoham, Y., & Tennenholtz, M. (1997). On the emergence of social conventions: Modeling, analysis, and simulations. *Artificial Intelligence*, *94*, 139–166.
38. Steels, L. (1997). The synthetic modeling of language origins. *Evolution of Communication*, *1*(1), 1–34.
39. Traag, V. (2011). Indirect reciprocity through gossiping can lead to cooperative clusters. In *IEEE ALIFE* (pp. 154–161).
40. Urbano, P., Ja, B., Antunes, L., & Moniz, L. (2009). Force versus majority: A comparison in convention emergence efficiency. In *Coordination, Organizations, Institutions and Norms in Agent Systems IV* (pp. 48–63).
41. Villatoro, D., Sabater-Mir, J., & Sen, S. (2011a). Social Instruments for Robust Convention Emergence. In *Twenty-Second International Joint Conference On Artificial Intelligence (IJCAI)* (p. 6). Barcelona, Spain.
42. Villatoro, D., Sen, S., & Sabater-Mir, J. (2011b). Exploring the dimensions of convention emergence in multiagent systems. *Advances in Complex Systems*, *14*(02), 201–227.
43. Ye, W., Heidemann, J., & Estrin, D. (2004). Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, *12*(3), 493–506.
44. Young, H. P. (1993). The evolution of conventions. *Econometrica*, *61*(1), 57–84.