A comprehensive survey of multi-agent reinforcement learning

By: L. Busoniu, R. Babuska, and B. De Schutter

Presentation by Christopher Gonzalez

Abstract

Abstract

- Multi-agent systems are rapidly finding applications in a variety of domains and the complexity of many tasks arising in these domains make them difficult to solve with preprogrammed agent behaviors, so the agents must instead discover a solution on their own using learning.
- A central issue in the field of multi-agent-reinforcement learning (MARL) is the formal statement of the multi-agent learning goal.
- Different viewpoints on this issue have led to the proposal of many different goals, two focal points can be distinguished: **stability of the agents' learning dynamics**, and **adaptation to the changing behavior of the other agents**.
- The MARL algorithms aim at one of these two goals or at a combination of both, in a fully cooperative, fully competitive, or more general setting
- A representative selection of these algorithms is discussed in detail in this paper, together with the specific issues that arise in each category. Additionally, the benefits and challenges of MARL are described along with some of the problem domains where MARL techniques have been applied

I. Introduction

I. Introduction - Introduction to MARL

- A multi-agent system can be defined as a group of autonomous, interacting entities sharing a common environment, which they perceive with sensors and upon which they act with actuators.
- Applications in: robotic teams, distributed control, resource management, collaborative decision support systems, data mining, etc
- Multi-agent systems may arise as the most natural way of looking at the system, or may provide an alternative perspective on systems that are originally regarded as centralized.
- For instance, in robotic teams, the control authority is naturally distributed among the robots. In resource management, while resources can be managed by a central authority, identifying each resource with an agent may provide a helpful, distributed perspective on the system.
- Although the agents in a multi-agent system can be programmed with behaviors designed in advance, it is often necessary that they learn new behaviors online, such that the performance of the agent or of the whole multi-agent system gradually improves
- In an environment that changes over time, a hardwired behavior may become unappropriate.

I. Introduction - Introduction to MARL

- A reinforcement learning (RL) agent learns by trial-and error interaction with its dynamic environment.
- At each time step, the agent perceives the complete state of the environment and takes an action, which causes the environment to transit into a new state.
- The agent receives a scalar reward signal that evaluates the quality of this transition.
- This feedback is less informative than in supervised learning, where the agent would be given the correct actions to take
- The RL feedback is, however, more informative than in unsupervised learning, where the agent would be left to discover the correct actions on its own, without any explicit feedback on its performance
- Well-understood algorithms with good convergence and consistency properties are available for solving the single-agent RL task, both when the agent knows the dynamics of the environment and the reward function (the task model), and when it does not.
- Together with the simplicity and generality of the setting, this makes RL attractive also for multi-agent learning.

I. Introduction – Challenges for RL in multi-agent systems

- The difficulty of defining a good learning goal for the multiple RL agents.
- Furthermore, most of the times each learning agent must keep track of the other learning (and therefore, nonstationary) agents.
- Only then will it be able to coordinate its behavior with theirs, such that a coherent joint behavior results.
- The nonstationarity also invalidates the convergence properties of most single-agent RL algorithms.
- In addition, the scalability of algorithms to realistic problem sizes, already problematic in single-agent RL, is an even greater cause for concern in MARL.

I. Introduction

- A wide variety of approaches to exploit MARL's benefits and address its challenges have been proposed over the last years.
- These approaches integrate developments in the areas of single-agent RL, game theory, and more general direct policy search techniques.
- The goal of this paper is to provide a comprehensive review of MARL.
- The authors select a representative set of approaches that allows them to identify the structure of the field, to provide insight into the current state of the art, and to determine some important directions for future research.

I. Introduction A. Contribution and related work

- This paper provides a detailed discussion of MARL techniques for fully cooperative, fully competitive, and mixed (neither cooperative nor competitive) tasks.
- The focus is placed on autonomous multiple agents learning how to solve dynamic tasks online, using learning techniques with roots in dynamic programming and temporal-difference RL.
- Different viewpoints on the central issue of the learning goal in MARL are discussed
- The authors provide an overview of the challenges and benefits in MARL, and of the problem domains where MARL techniques have been applied.
- They identify a set of important open issues and suggest promising directions to address these issues.

I. Introduction- A. Contribution and related work - Game Theory

- Besides single-agent RL, MARL has strong connections with game theory, evolutionary computation, and optimization theory.
- Game theory – the study of multiple interacting agents trying to maximize their rewards and especially the theory of learning in games, make an essential contribution to MARL.
- The authors focus on algorithms for dynamic multiagent tasks, whereas most game-theoretic results deal with static (stateless) one-shot or repeated tasks.
- The authors investigate the contribution of game theory to MARL algorithms for dynamic tasks and review relevant game-theoretic algorithms for static games.
- Other authors have investigated more closely the relationship between game theory and MARL.
- Bowling and Veloso [13] discuss several MARL algorithms, showing that these algorithms combine temporal-difference RL with game-theoretic solvers for the static games arising in each state of the dynamic environment.

I. Introduction- A. Contribution and related work - Evolutionary training

- Since the authors are interested in online techniques that exploit the special structure of the RL task by learning a value function, they do not review here evolutionary learning techniques.
- Evolutionary learning, and in general direct optimization of the agent behaviors, cannot readily benefit from the RL task structure.
- Evolutionary game theory sits at the intersection of evolutionary learning and game theory.
- The authors discuss only the contribution of evolutionary game theory to the analysis of multi-agent RL dynamics.

I. Introduction- B. Overview

- The next section introduces the necessary background in single-agent and multi-agent RL.
- Section III reviews the main benefits of MARL and the most important challenges that arise in the field, among which is the definition of an appropriate formal goal for the learning multi-agent system.
- Section IV discusses the formal goals put forward in the literature, which consider stability of the agent's learning process and adaptation to the dynamic behavior of the other agents.
- Section V provides a taxonomy of MARL techniques.
- Section VI reviews a representative selection of MARL algorithms, grouping them by the type of targeted learning goal (stability, adaptation, or a combination of both) and by the type of task (fully cooperative, fully competitive, or mixed).
- Section VII then gives a brief overview of the problem domains where MARL has been applied.
- Section VIII distills an outlook for the MARL field, consisting of important open questions and some suggestions for future research.
- Section IX concludes and closes the paper

II. BACKGROUND: REINFORCEMENT LEARNING

II. BACKGROUND: REINFORCEMENT LEARNING

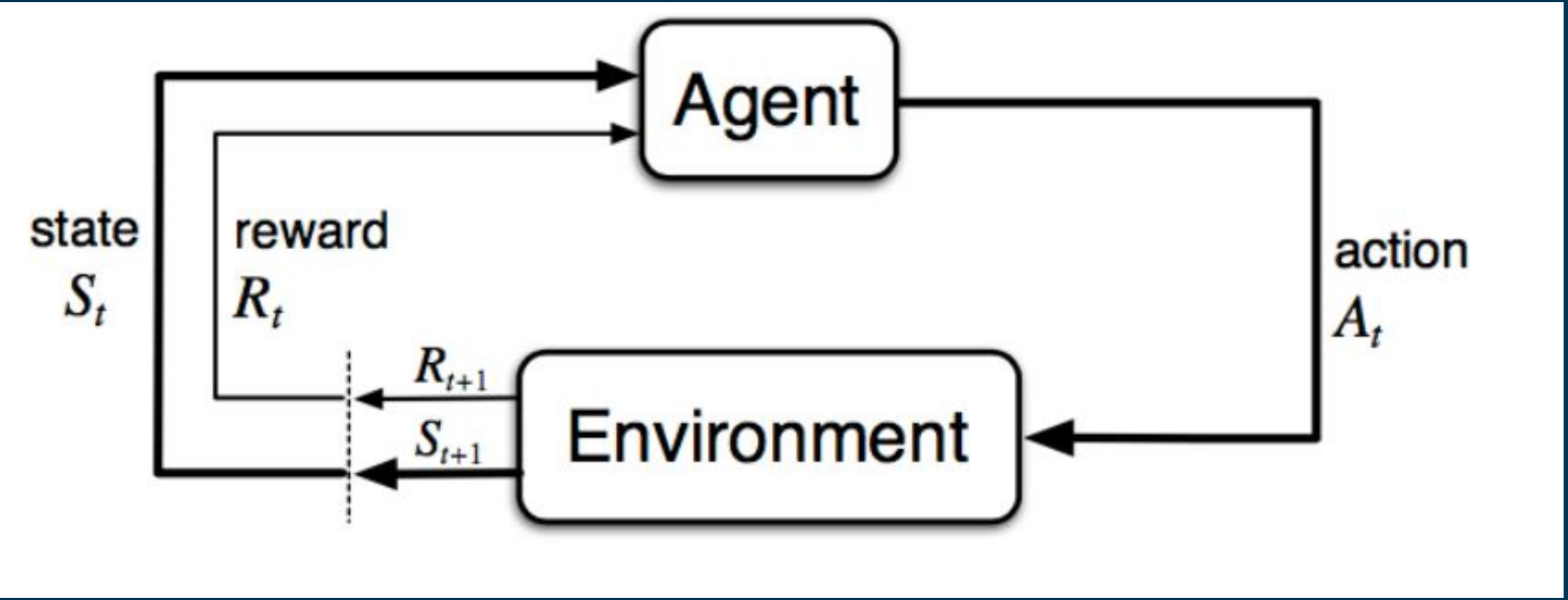
- In this section, the necessary background on single-agent and multi-agent RL is introduced
- First, the single agent task is defined, and its solution is characterized.
- Then, the multi-agent task is defined.
- Static multi-agent tasks are introduced separately, together with necessary game-theoretic concepts.
- The discussion is restricted to finite state and action spaces, as most MARL results is given for finite spaces.

A. The single-agent case

- In single-agent RL, the environment of the agent is described by a Markov decision process
- Definition 1: A finite Markov decision process is a tuple $\{X, U, f, \rho\}$ where
- X is the finite set of environment states,
- U is the finite set of agent actions,
- $f : X \times U \times X \rightarrow [0, 1]$ is the state transition probability function, and
- $\rho : X \times U \times X \rightarrow R$ is the reward function.

- Throughout the paper, the standard control-theoretic notation is used:
- x for state,
- X for state space,
- u for control action,
- U for action space,
- f for environment (process) dynamics.
- The authors denote reward functions by ρ, to distinguish them from the instantaneous rewards r and the returns R.
- They denote agent policies by h.

A. The single-agent case – Markov Decision Process



A. The single-agent case

- The state signal $x_k \in X$ describes the environment at each discrete time step k. The agent can alter the state at each time step by taking actions $u_k \in U$
- As a result of action uk, the environment changes state from x to some $x_{k+1} \in X$ according to the state transition probabilities given by f: the probability of ending up in xk+1 given that uk is executed in xk is:

$$f(x_k, u_k, x_{k+1})$$

- The agent receives a scalar reward $r_{k+1} \in \mathbb{R}$ according to the reward function ρ:

$$r_{k+1} = \rho(x_k, u_k, x_{k+1})$$

- This reward evaluates the immediate effect of action uk, i.e., the transition from xk to xk+1.
- It says, however, nothing directly about the long-term effects of this action.

A. The single-agent case – Agent behaviors

- The behavior of the agent is described by its policy h, which specifies how the agent chooses its actions given the state.
- The policy may be either stochastic,

$$h : X \times U \rightarrow [0, 1]$$

- or deterministic

$$\overline{h} : X \rightarrow U$$

- A policy is called stationary if it does not change over time.

A. The single-agent case – Agent behaviors

- The agent's goal is to maximize, at each time step k, the expected discounted return:
- where γ ∈ [0, 1) is the discount factor, and the expectation is taken over the probabilistic state transitions.
- The quantity Rk compactly represents the reward accumulated by the agent in the long run.

$$R_k = E\left\{\sum_{j=0}^{\infty} \gamma^j r_{k+j+1}\right\}$$

- (1)
- Other possibilities of defining the return exist.
- The discount factor γ can be regarded as encoding increasing uncertainty about rewards that will be received in the future, or as a means to bound the sum which otherwise might grow infinitely

A. The single-agent case – Q function

- The task of the agent is therefore to maximize its long term performance, while only receiving feedback about its immediate, one-step performance.
- One way it can achieve this is by computing an optimal action-value function.
- The action-value function (Q-function), $Q^h : X \times U \to \mathbb{R}$ is the expected return of a state-action pair given the policy h: $Q^h(x,u) = E\left\{\sum_{j=0}^{\infty} \gamma^j r_{k+j+1} \,|\, x_k = x, u_k = u, h\right\}$
- The optimal Q-function is defined as (2)

$$Q^*(x,u) = \sum_{x' \in X} f(x,u,x')\left[\rho(x,u,x') + \gamma \max_{u'} Q^*(x',u')\right]$$

- This equation states that the optimal value of taking u in x is the expected immediate reward plus the expected (discounted) optimal value attainable from the next state (the expectation is explicitly written as a sum since X is finite).
- The greedy policy is deterministic and picks for every state the action with the highest Q-value:

$$\overline{h}(x) = \arg\max_u Q(x,u)$$

A. The single-agent case – Q-learning

- Many single-agent RL algorithms exist such as:
- Model-based methods based on dynamic programming
- Model-free methods based on online estimation of value functions and
- Model-learning methods that estimate a model, and then learn using model-based techniques.
- Most MARL algorithms are derived from a model-free algorithm called **Q-learning**

- Q-learning turns (2) into an iterative approximation procedure.
- The current estimate of $Q*$ is updated using estimated samples of the right-hand side of (2).
- These samples are computed using actual experience with the task, in the form of rewards rk+1 and pairs of subsequent states xk, xk+1:

$$Q_{k+1}(x_k, u_k) = Q_k(x_k, u_k) + \alpha_k \left[r_{k+1} + \gamma \max_{u'} Q_k(x_{k+1}, u') - Q_k(x_k, u_k) \right] \quad (4)$$

A. The single-agent case – Q-learning

- Since (4) does not require knowledge about the transition and reward functions, Q-learning is model-free.
- The learning rate $\alpha_k \in (0, 1]$ specifies how far the current estimate $Q_k(x_k, u_k)$ is adjusted towards the update target (sample) $r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u')$
- The learning rate is typically time-varying, decreasing with time.
- Separate learning rates may be used for each state-action pair.
- The expression inside the square brackets is the temporal difference, i.e., the difference between estimates of $Q(x_k, u_k)$ at two successive time steps, k + 1 and k

A. The single-agent case – Q-learning

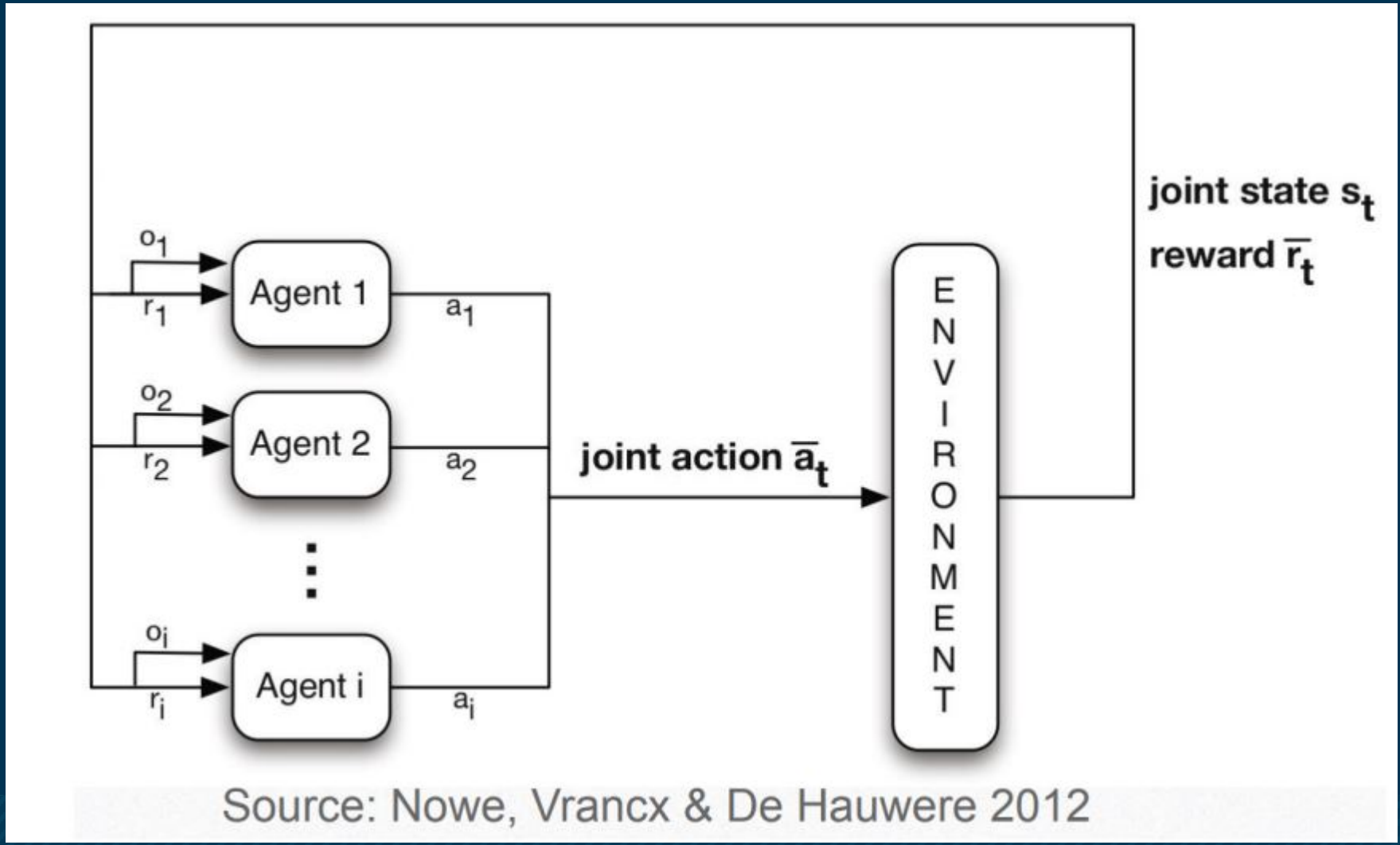
- The sequence Qk provably converges to $Q*$ under the following conditions:
- Explicit, distinct values of the Q-function are stored and updated for each state-action pair.
- The time series of learning rates used for each state-action pair sums to infinity, whereas the sum of its squares is finite.
- The agent keeps trying all actions in all states with nonzero probability.

B. The multi-agent case

- The generalization of the Markov decision process to the multi-agent case is the stochastic game
- A stochastic game (SG) is a tuple {X, U1, . . . , Un, f, ρ1, . . . , ρn} where
- n is the number of agents,
- X is the discrete set of environment states,
- Ui , i = 1, . . . , n are the discrete sets of actions available to the agents, yielding the joint action set U = U1 × · · · × Un, f : X × U × X → [0, 1] is the state transition probability function, and
- ρi : X × U × X → R, i = 1, . . . , n are the reward functions of the agents

- In the multi-agent case, the state transitions are the result of the joint action of all the agents
- Consequently, the rewards ri,k+1 and the returns Ri,k also depend on the joint action
- The policies hi : X × Ui → [0, 1] form together the joint policy h.
- The Q-function of each agent depends on the joint action and is conditioned on the joint policy

B. The multi-agent case



Source: Nowe, Vrancx & De Hauwere 2012

B. The multi-agent case

- If $\rho 1 = \cdots = \rho n$, all the agents have the same goal (to maximize the same expected return), and the SG is fully cooperative.
- If $n = 2$ and $\rho 1 = -\rho 2$, the two agents have opposite goals, and the SG is fully competitive.
- Mixed games are stochastic games that are neither fully cooperative nor fully competitive.

C. Static, repeated, and stage games

- Many MARL algorithms are designed for static (stateless) games, or work in a stage-wise fashion, looking at the static games that arise in each state of the stochastic game.
- Some game-theoretic definitions and concepts regarding static games are therefore necessary to understand these algorithms
- A static (stateless) game is a stochastic game with $X = \emptyset$. (empty state set)
- Since there is no state signal, the rewards depend only on the joint actions ρi : $U \rightarrow R$
- When there are only two agents, the game is often called a bimatrix game, because the reward function of each of the two agents can be represented as a $|U1| \times |U2|$ matrix with the rows corresponding to the actions of agent 1, and the columns to the actions of agent 2, where $|\cdot|$ denotes set cardinality.
- Fully competitive static games are also called zero-sum games, because the sum of the agents' reward matrices is a zero matrix.
- Mixed static games are also called general-sum games, because there is no constraint on the sum of the agents' rewards.

C. Static, repeated, and stage games

- When played repeatedly by the same agents, the static game is called a repeated game.
- The main difference from a one-shot game is that the agents can use some of the game iterations to gather information about the other agents or the reward functions, and make more informed decisions thereafter.
- A stage game is the static game that arises when the state of an SG is fixed to some value.
- The reward functions of the stage game are the expected returns of the SG when starting from that particular state.
- Since in general the agents visit the same state of an SG multiple times, the stage game is a repeated game.
- In a static or repeated game, the policy loses the state argument and transforms into a strategy $\sigma i : Ui \rightarrow [0, 1]$.
- An agent's strategy for the stage game arising in some state of the SG is its policy conditioned on that state value.
- MARL algorithms relying on the stage-wise approach learn strategies separately for every stage game.
- The agent's overall policy is then the aggregate of these strategies.

C. Static, repeated, and stage games

- Stochastic strategies (and consequently, stochastic policies) are of a more immediate importance in MARL than in single-agent RL, because in certain cases, like for the Nash equilibrium, the solutions can only be expressed in terms of stochastic strategies
- An important solution concept for static games, is the Nash equilibrium.

- First, define the best response of agent i to a vector of opponent strategies as the strategy σ ∗ i that achieves the maximum expected reward given these opponent strategies:

$$E\{r_i|\sigma_1,\ldots,\sigma_i,\ldots,\sigma_n\} \leq E\{r_i|\sigma_1,\ldots,\sigma_i^*,\ldots,\sigma_n\} \quad \forall\sigma_i \quad (6)$$

C. Static, repeated, and stage games- Nash Equilibrium

- A Nash equilibrium is a joint strategy [σ∗1, . . . , σ∗n] T such that each individual strategy σ∗i is a best-response to the others
- The Nash equilibrium describes a status quo, where no agent can benefit by changing its strategy as long as all other agents keep their strategies constant.
- Any static game has at least one (possibly stochastic) Nash equilibrium; some static games have multiple Nash equilibria.
- Nash equilibria are used by many MARL algorithms reviewed in the sequel, either as learning goal, or both as learning goal and directly in the update rules.

Thank You