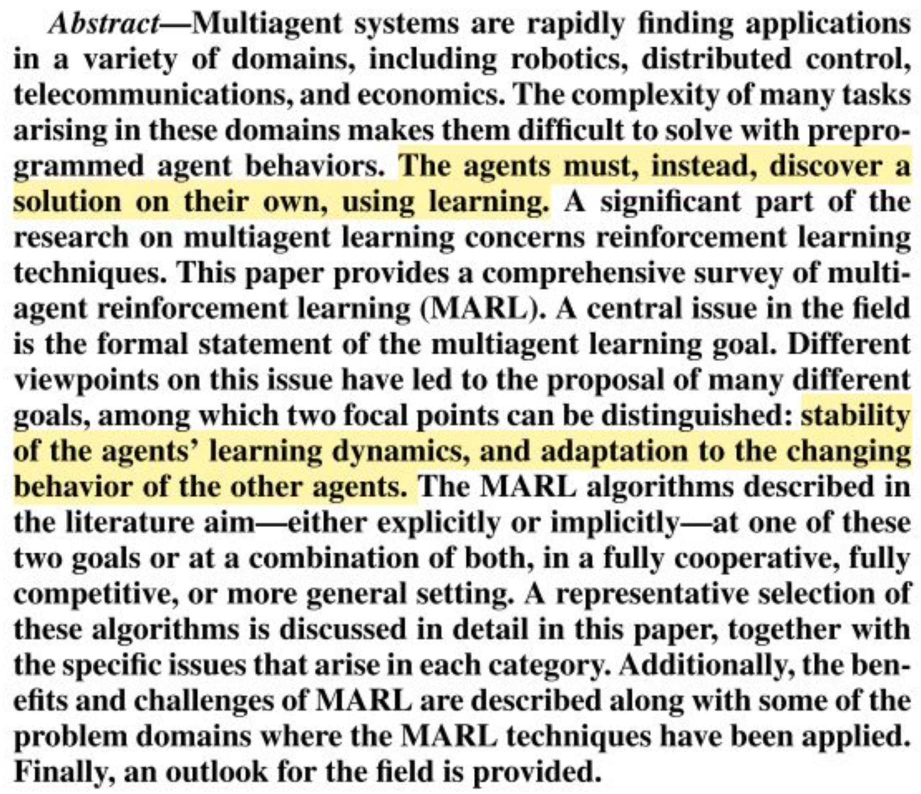
A Comprehensive Survey of Multi-Agent Reinforcement Learning

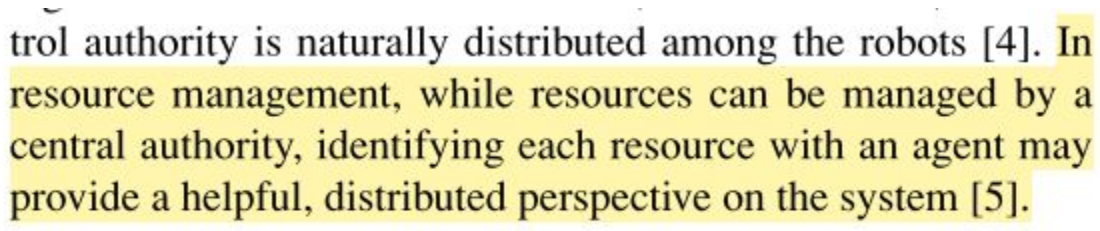
Abstract



Abstract—Multiagent systems are rapidly finding applications in a variety of domains, including robotics, distributed control, telecommunications, and economics. The complexity of many tasks arising in these domains makes them difficult to solve with preprogrammed agent behaviors. The agents must, instead, discover a solution on their own, using learning. A significant part of the research on multiagent learning concerns reinforcement learning techniques. This paper provides a comprehensive survey of multiagent reinforcement learning (MARL). A central issue in the field is the formal statement of the multiagent learning goal. Different viewpoints on this issue have led to the proposal of many different goals, among which 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 described in the literature aim—either explicitly or implicitly—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 the MARL techniques have been applied. Finally, an outlook for the field is provided.

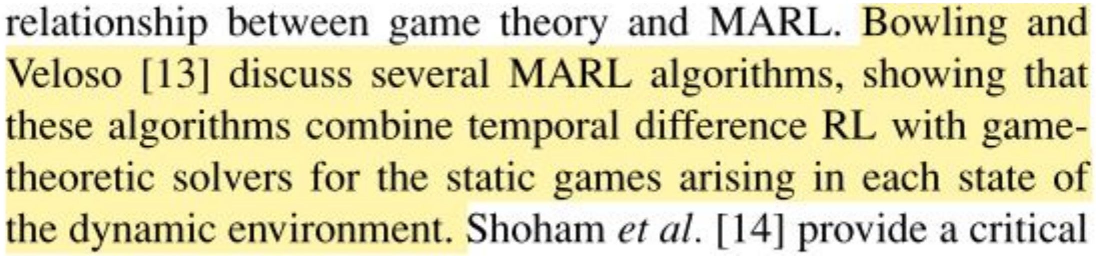
Introduction

- Central System
- Distributed System


trol authority is naturally distributed among the robots [4]. 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 [5].

Related Work

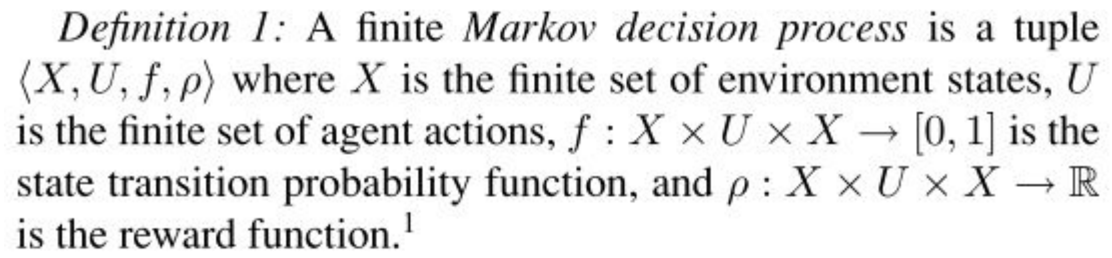
- MARL is highly related to Game Theory (Multi-Agent Systems).
- The main difference is:
  - MARL focuses on dynamic multi-agent tasks, whereas,
  - Game theory deals with static (stateless) one-shot or repeated tasks.



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. Shoham *et al*. [14] provide a critical

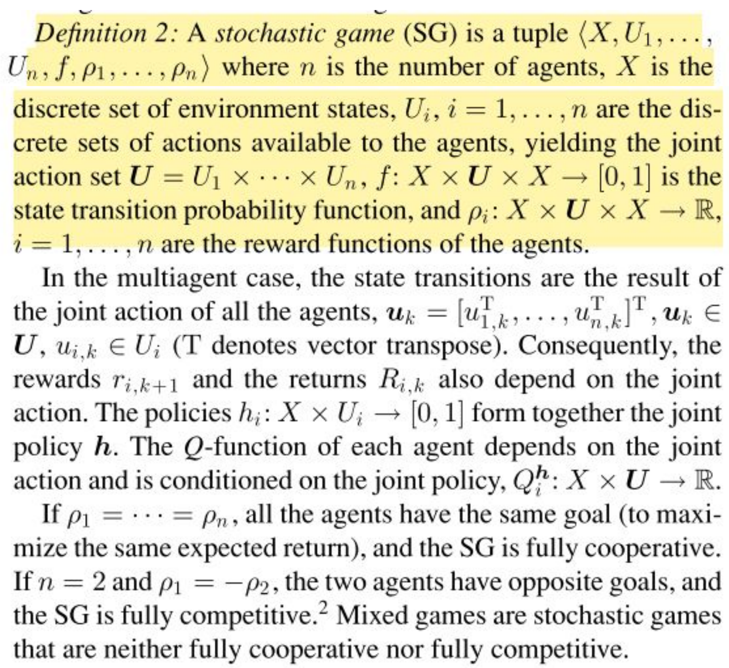
Background: Reinforcement Learning

A. Single Agent Case
   a. Model based
   b. Model free
   c. Q learning
   d. Exploration and exploitation
      i. Epsilon-greedy
      ii. Boltzmann exploration strategy*



Definition 1: A finite *Markov decision process* is a tuple $\langle X, U, f, \rho \rangle$ where X is the finite set of environment states, U is the finite set of agent actions, $f : X \times U \times X \to [0, 1]$ is the state transition probability function, and $\rho : X \times U \times X \to \mathbb{R}$ is the reward function.[1]

A. Single Agent Case
B. Multi-Agent Case
    a. The generalization of Markov Decision Process in multi-agent case is the Stocastic Game.

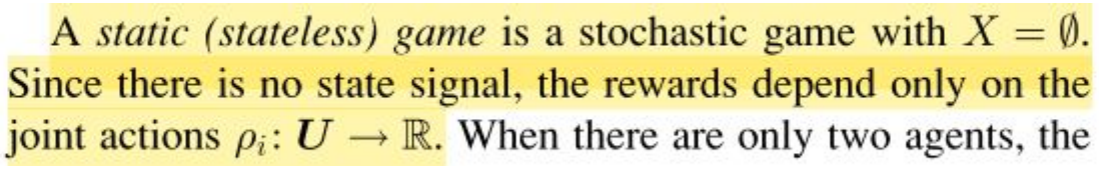


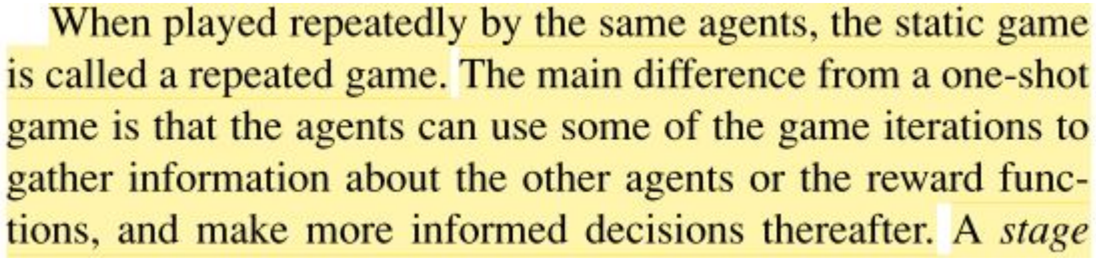
Definition 2: A *stochastic game* (SG) is a tuple $\langle X, U_1, \ldots, U_n, f, \rho_1, \ldots, \rho_n \rangle$ where n is the number of agents, X is the discrete set of environment states, U_i, $i = 1, \ldots, n$ are the discrete sets of actions available to the agents, yielding the joint action set $\boldsymbol{U} = U_1 \times \cdots \times U_n$, $f \colon X \times \boldsymbol{U} \times X \to [0, 1]$ is the state transition probability function, and $\rho_i \colon X \times \boldsymbol{U} \times X \to \mathbb{R}$, $i = 1, \ldots, n$ are the reward functions of the agents.

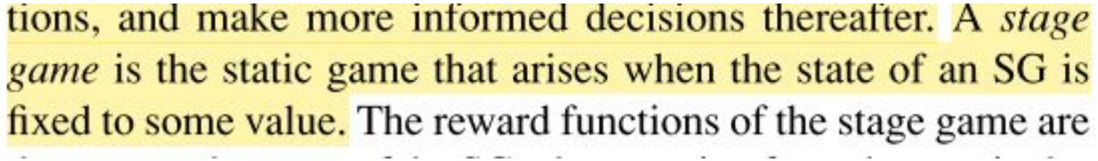
In the multiagent case, the state transitions are the result of the joint action of all the agents, $\boldsymbol{u}_k = [u_{1,k}^{\mathrm{T}}, \ldots, u_{n,k}^{\mathrm{T}}]^{\mathrm{T}}$, $\boldsymbol{u}_k \in \boldsymbol{U}$, $u_{i,k} \in U_i$ (T denotes vector transpose). Consequently, the rewards $r_{i,k+1}$ and the returns $R_{i,k}$ also depend on the joint action. The policies $h_i \colon X \times U_i \to [0, 1]$ form together the joint policy $\boldsymbol{h}$. The *Q*-function of each agent depends on the joint action and is conditioned on the joint policy, $Q_i^{\boldsymbol{h}} \colon X \times \boldsymbol{U} \to \mathbb{R}$.

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.[2] Mixed games are stochastic games that are neither fully cooperative nor fully competitive.

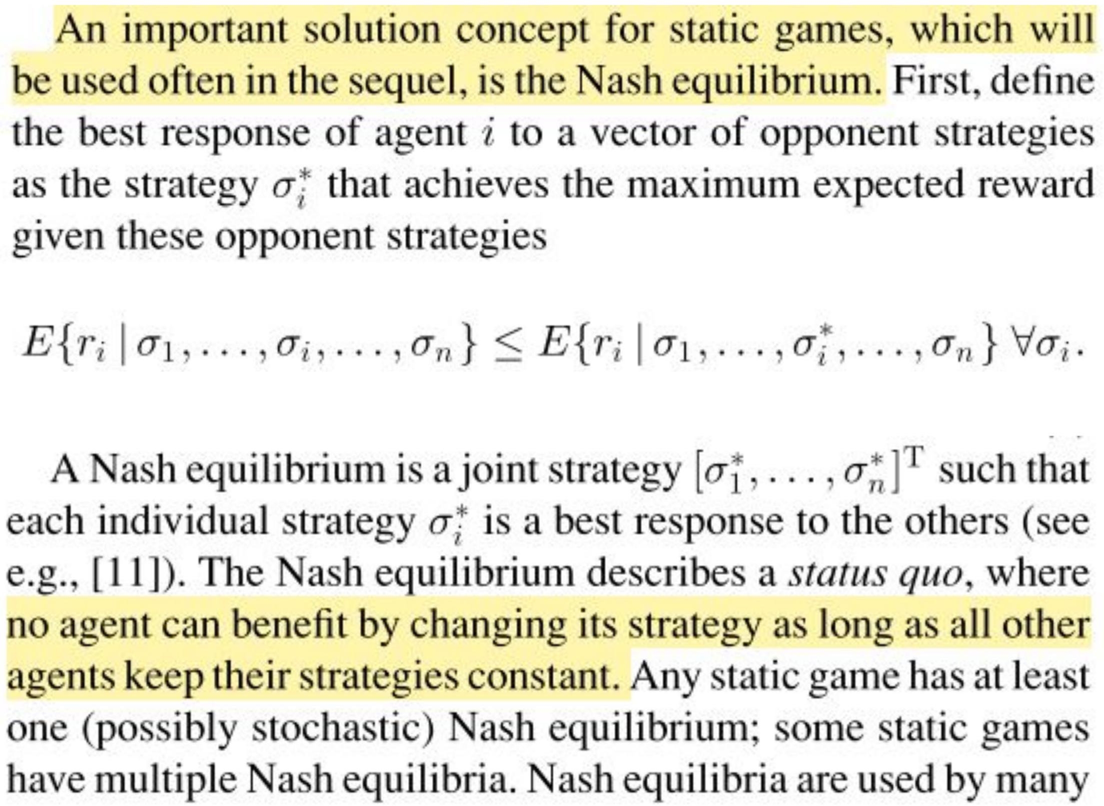
A. Single Agent Case
B. Multi Agent Case
C. Static, Repeated and Stage games


A *static (stateless) game* is a stochastic game with $X = \emptyset$. Since there is no state signal, the rewards depend only on the joint actions $\rho_i \colon \boldsymbol{U} \rightarrow \mathbb{R}$. When there are only two agents, the


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*


tions, 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

Nash Equilibrium



An important solution concept for static games, which will be used often in the sequel, 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

$$E\{r_i \,|\, \sigma_1, \ldots, \sigma_i, \ldots, \sigma_n\} \leq E\{r_i \,|\, \sigma_1, \ldots, \sigma_i^*, \ldots, \sigma_n\} \; \forall \sigma_i .$$

A Nash equilibrium is a joint strategy $[\sigma_1^*, \ldots, \sigma_n^*]^{\mathrm{T}}$ such that each individual strategy σ_i^* is a best response to the others (see e.g., [11]). 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

Benefits of MARL

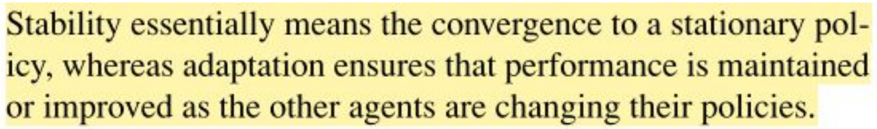
- Parallel computing
- Experience sharing
- Robustness (inherent)
- High Scalability

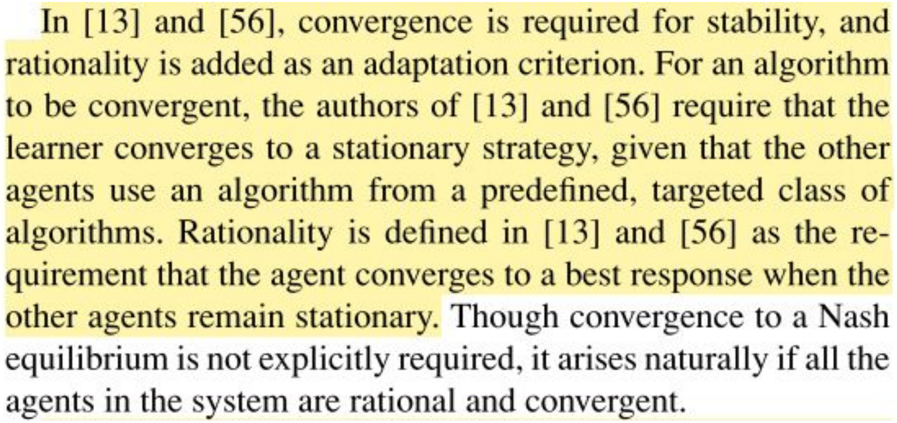
Challenges in MARL

- The curse of dimensionality
- Specifying a goal
- Nonstationarity
- Exploration-exploitation trade off
- Coordination

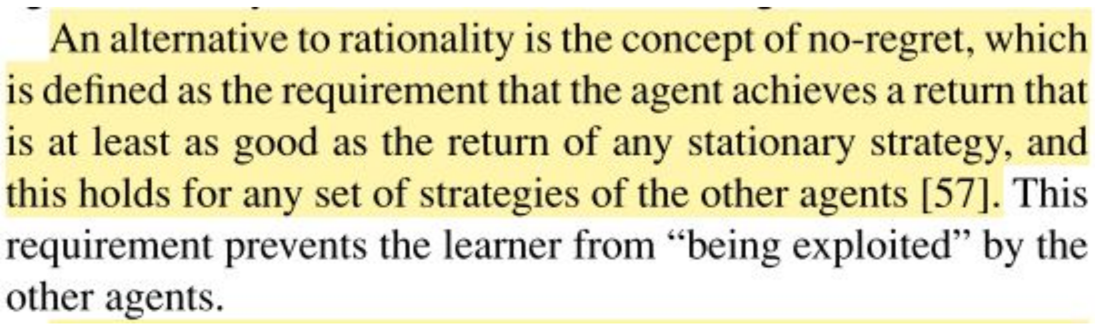
MARL Learning Goal

- Stability of learning dynamics
- Adaptation to the dynamic behaviour of the other agents


Stability essentially means the convergence to a stationary policy, whereas adaptation ensures that performance is maintained or improved as the other agents are changing their policies.

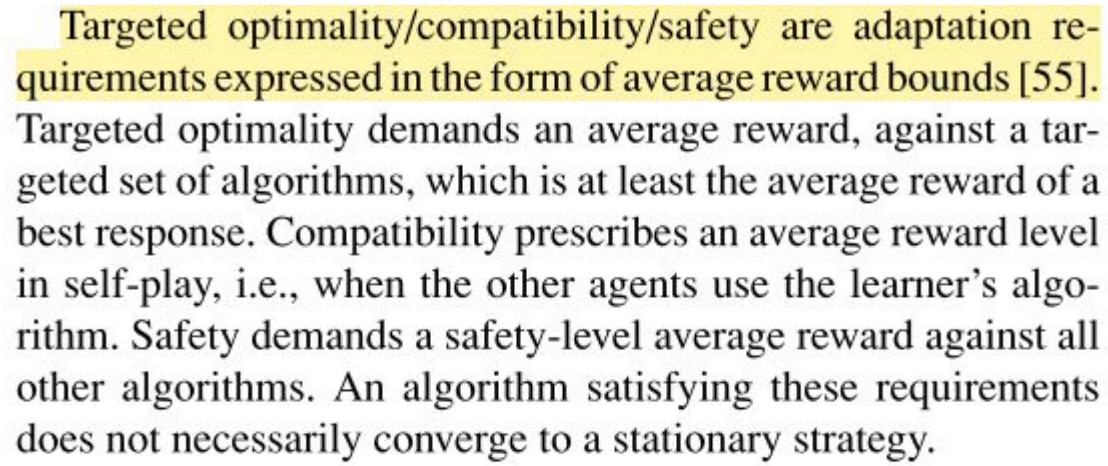

In [13] and [56], convergence is required for stability, and rationality is added as an adaptation criterion. For an algorithm to be convergent, the authors of [13] and [56] require that the learner converges to a stationary strategy, given that the other agents use an algorithm from a predefined, targeted class of algorithms. Rationality is defined in [13] and [56] as the requirement that the agent converges to a best response when the other agents remain stationary. Though convergence to a Nash equilibrium is not explicitly required, it arises naturally if all the agents in the system are rational and convergent.

No regret: alternative to rationality (adaptation)



An alternative to rationality is the concept of no-regret, which is defined as the requirement that the agent achieves a return that is at least as good as the return of any stationary strategy, and this holds for any set of strategies of the other agents [57]. This requirement prevents the learner from "being exploited" by the other agents.

Targeted optimality/compatibility/safety



Targeted optimality/compatibility/safety are adaptation requirements expressed in the form of average reward bounds [55]. Targeted optimality demands an average reward, against a targeted set of algorithms, which is at least the average reward of a best response. Compatibility prescribes an average reward level in self-play, i.e., when the other agents use the learner's algorithm. Safety demands a safety-level average reward against all other algorithms. An algorithm satisfying these requirements does not necessarily converge to a stationary strategy.

Stability and Adaptation in MARL

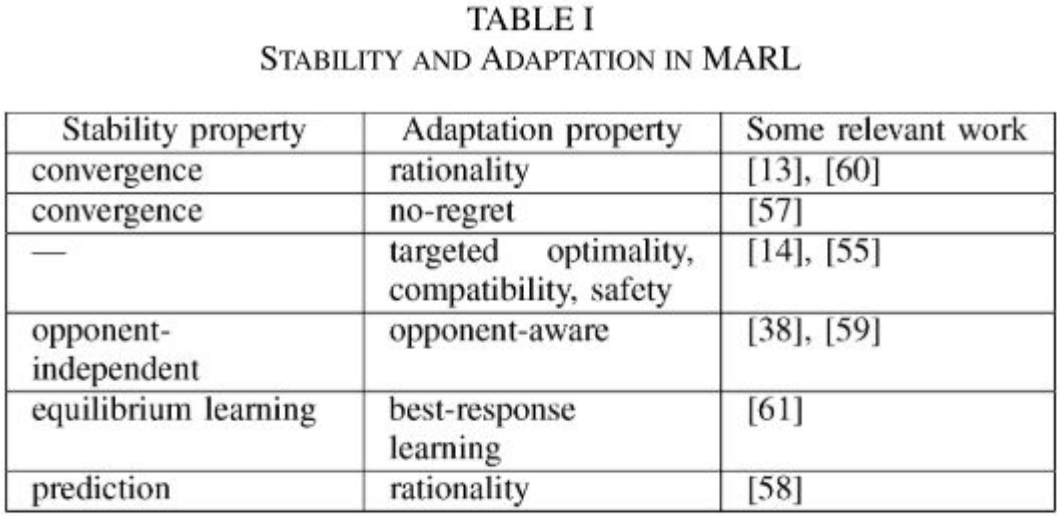
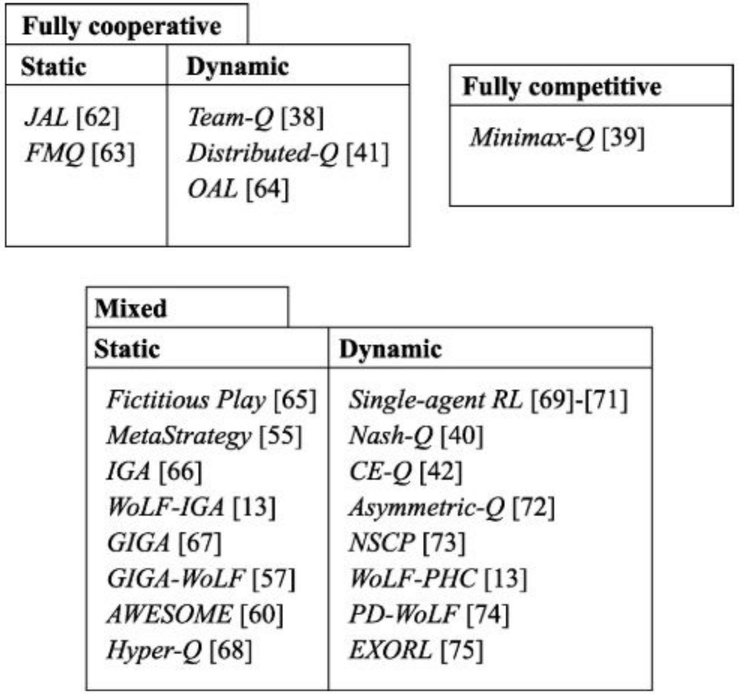


TABLE I
STABILITY AND ADAPTATION IN MARL

Stability property	Adaptation property	Some relevant work
convergence	rationality	[13], [60]
convergence	no-regret	[57]
—	targeted optimality, compatibility, safety	[14], [55]
opponent-independent	opponent-aware	[38], [59]
equilibrium learning	best-response learning	[61]
prediction	rationality	[58]

Taxonomy of MARL Algorithms

- Task Based
  - Fully cooperative,
  - Fully competitive,
  - Mixed



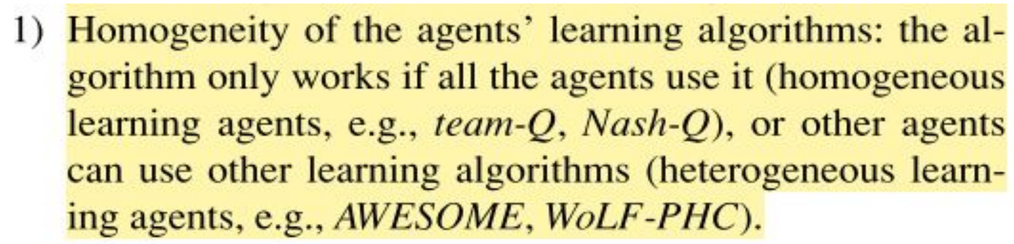
Fully cooperative	
Static	**Dynamic**
JAL [62]	*Team-Q* [38]
FMQ [63]	*Distributed-Q* [41]
	OAL [64]

Fully competitive
Minimax-Q [39]

Mixed	
Static	**Dynamic**
Fictitious Play [65]	*Single-agent RL* [69]-[71]
MetaStrategy [55]	*Nash-Q* [40]
IGA [66]	*CE-Q* [42]
WoLF-IGA [13]	*Asymmetric-Q* [72]
GIGA [67]	*NSCP* [73]
GIGA-WoLF [57]	*WoLF-PHC* [13]
AWESOME [60]	*PD-WoLF* [74]
Hyper-Q [68]	*EXORL* [75]

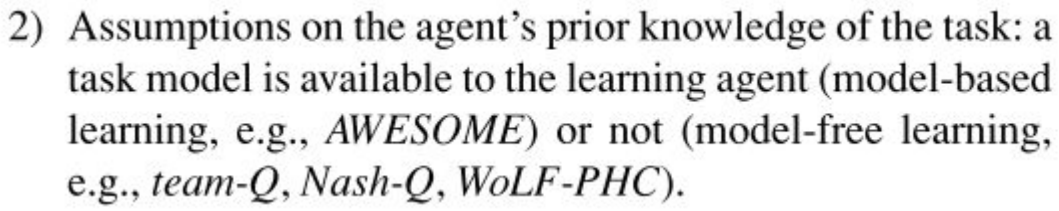
Fig. 1. Breakdown of MARL algorithms by the type of task they address.

- Homogeneity
  - Homogeneous
  - Heterogeneous



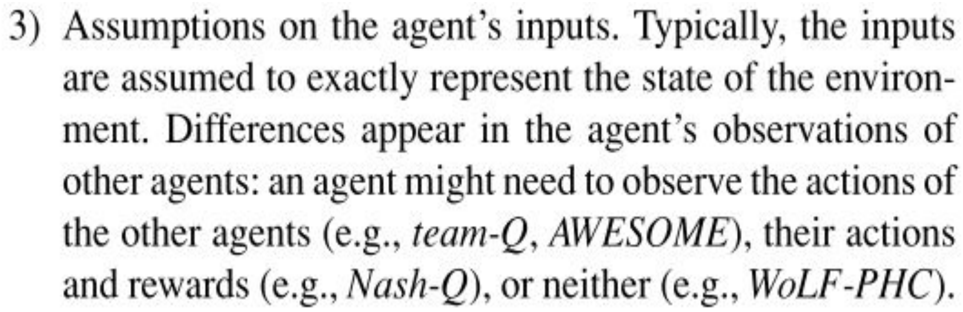
1) Homogeneity of the agents' learning algorithms: the algorithm only works if all the agents use it (homogeneous learning agents, e.g., *team-Q*, *Nash-Q*), or other agents can use other learning algorithms (heterogeneous learning agents, e.g., *AWESOME*, *WoLF-PHC*).

- Assumption on the agent's prior knowledge of the task:
  - Model based
  - Model free



2) Assumptions on the agent's prior knowledge of the task: a task model is available to the learning agent (model-based learning, e.g., *AWESOME*) or not (model-free learning, e.g., *team-Q*, *Nash-Q*, *WoLF-PHC*).

- Assumption on the agent's inputs: An agent might need to observe:
  - Actions of other agents
  - Actions and rewards
  - Neither



3) Assumptions on the agent's inputs. Typically, the inputs are assumed to exactly represent the state of the environment. Differences appear in the agent's observations of other agents: an agent might need to observe the actions of the other agents (e.g., *team-Q*, *AWESOME*), their actions and rewards (e.g., *Nash-Q*), or neither (e.g., *WoLF-PHC*).

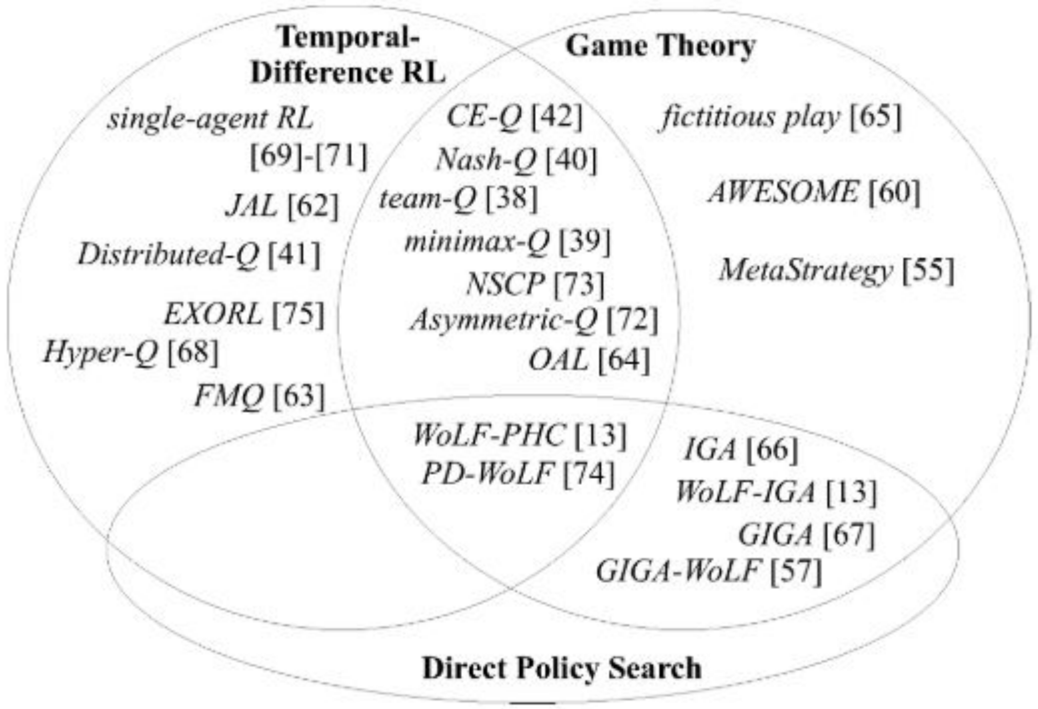
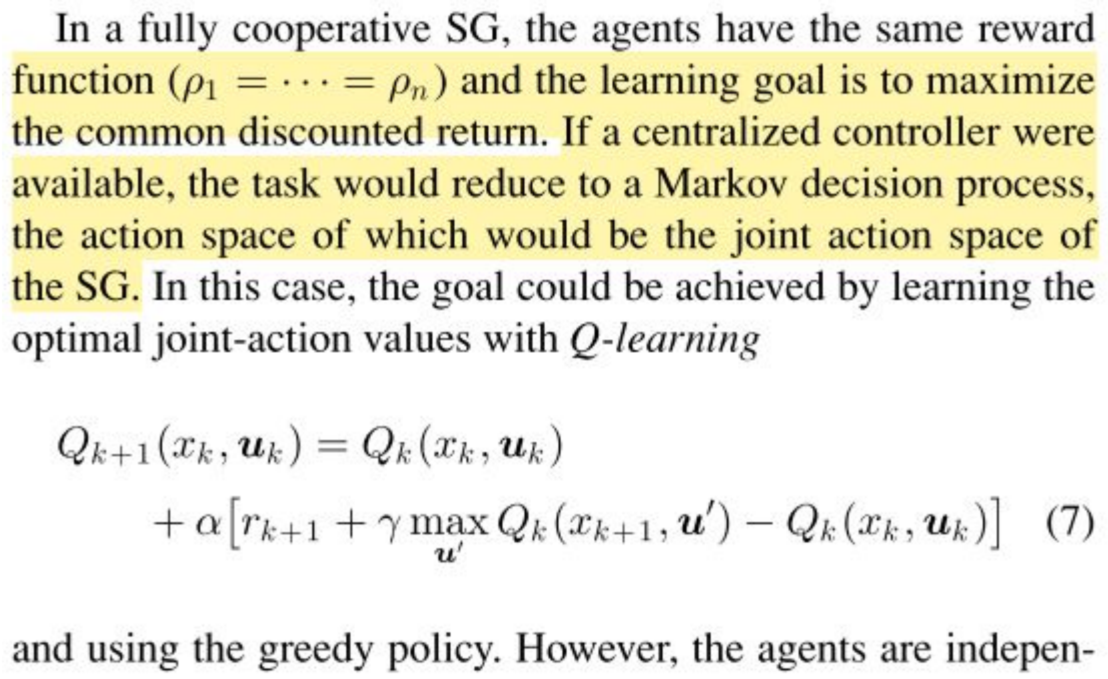


Fig. 2. MARL encompasses temporal-difference reinforcement learning, game theory, and direct policy search techniques.

MARL Algorithms

Fully Cooperative tasks

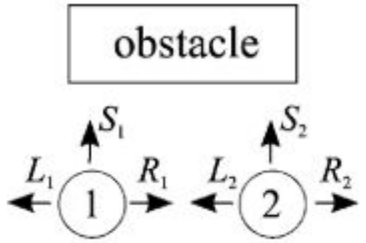


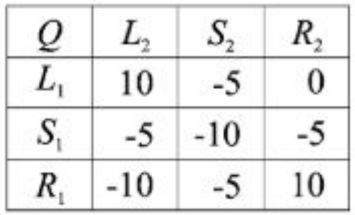
In a fully cooperative SG, the agents have the same reward function ($\rho_1 = \cdots = \rho_n$) and the learning goal is to maximize the common discounted return. If a centralized controller were available, the task would reduce to a Markov decision process, the action space of which would be the joint action space of the SG. In this case, the goal could be achieved by learning the optimal joint-action values with *Q-learning*

$$Q_{k+1}(x_k, \boldsymbol{u}_k) = Q_k(x_k, \boldsymbol{u}_k) + \alpha\left[r_{k+1} + \gamma \max_{\boldsymbol{u}'} Q_k(x_{k+1}, \boldsymbol{u}') - Q_k(x_k, \boldsymbol{u}_k)\right] \quad (7)$$

and using the greedy policy. However, the agents are indepen-

Need for coordination (Example 1)



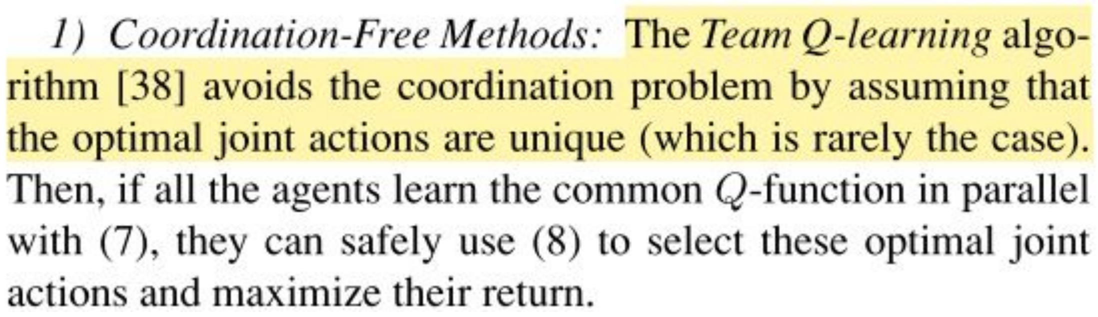


Q	L_2	S_2	R_2
L_1	10	-5	0
S_1	-5	-10	-5
R_1	-10	-5	10

Fig. 3. (Left) Two mobile agents approaching an obstacle need to coordinate their action selection. (Right) The common Q-values of the agents for the state depicted to the left.

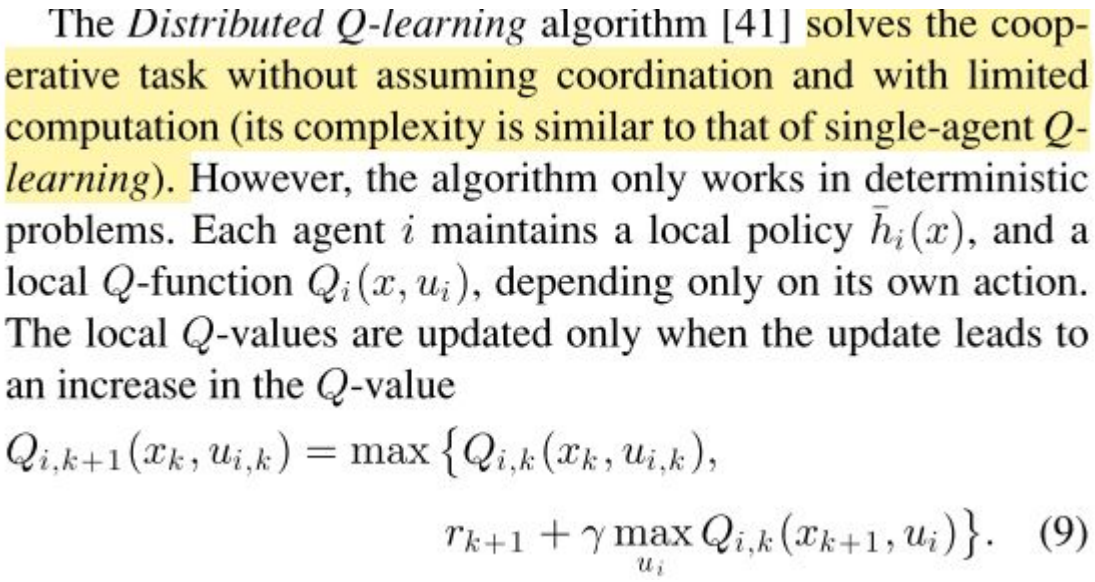
Coordination free methods

Team Q Learning



1) Coordination-Free Methods: The *Team Q-learning* algorithm [38] avoids the coordination problem by assuming that the optimal joint actions are unique (which is rarely the case). Then, if all the agents learn the common Q-function in parallel with (7), they can safely use (8) to select these optimal joint actions and maximize their return.

Distributed Q Learning

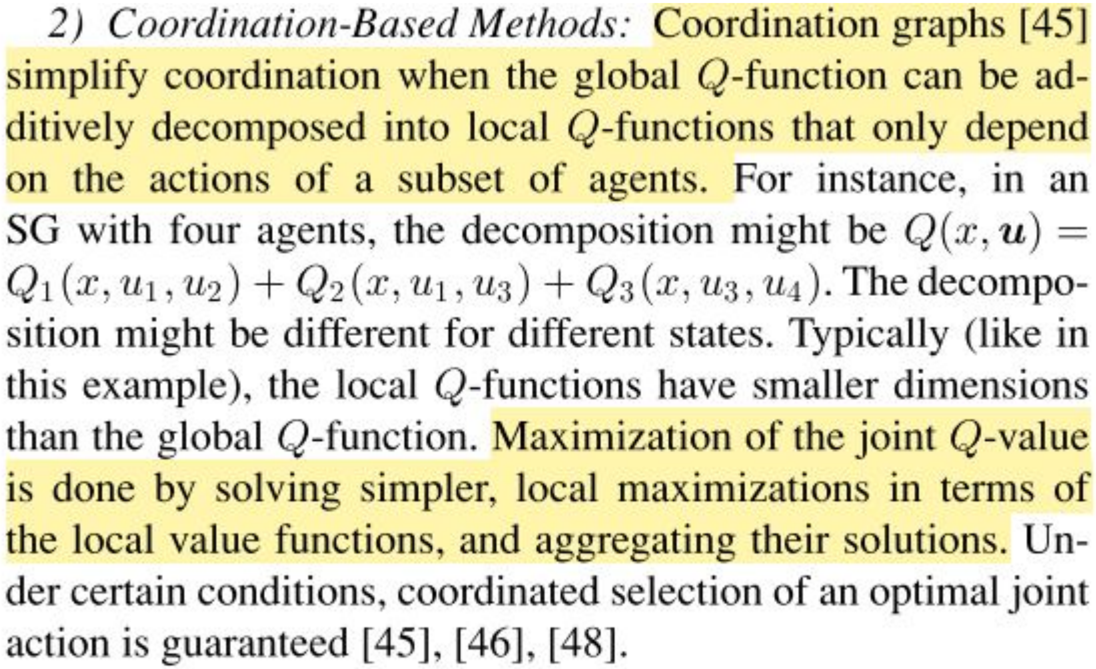


The *Distributed Q-learning* algorithm [41] solves the cooperative task without assuming coordination and with limited computation (its complexity is similar to that of single-agent *Q-learning*). However, the algorithm only works in deterministic problems. Each agent i maintains a local policy $\bar{h}_i(x)$, and a local Q-function $Q_i(x, u_i)$, depending only on its own action. The local Q-values are updated only when the update leads to an increase in the Q-value

$$Q_{i,k+1}(x_k, u_{i,k}) = \max\Big\{Q_{i,k}(x_k, u_{i,k}), \\ r_{k+1} + \gamma \max_{u_i} Q_{i,k}(x_{k+1}, u_i)\Big\}. \quad (9)$$

The local policy is updated only if the update leads to an improvement in the Q-values

Coordination Based methods

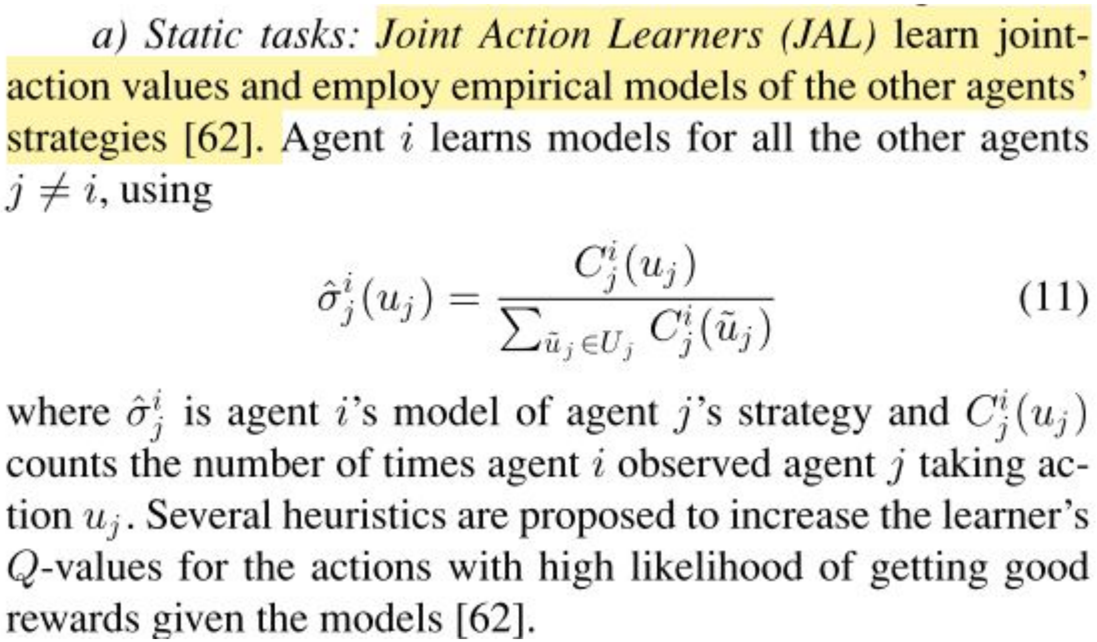


2) *Coordination-Based Methods:* Coordination graphs [45] simplify coordination when the global Q-function can be additively decomposed into local Q-functions that only depend on the actions of a subset of agents. For instance, in an SG with four agents, the decomposition might be $Q(x, \boldsymbol{u}) = Q_1(x, u_1, u_2) + Q_2(x, u_1, u_3) + Q_3(x, u_3, u_4)$. The decomposition might be different for different states. Typically (like in this example), the local Q-functions have smaller dimensions than the global Q-function. Maximization of the joint Q-value is done by solving simpler, local maximizations in terms of the local value functions, and aggregating their solutions. Under certain conditions, coordinated selection of an optimal joint action is guaranteed [45], [46], [48].

Indirect Coordination methods

They bias action selection towards actions that are likely to result in good rewards or returns.

JAL

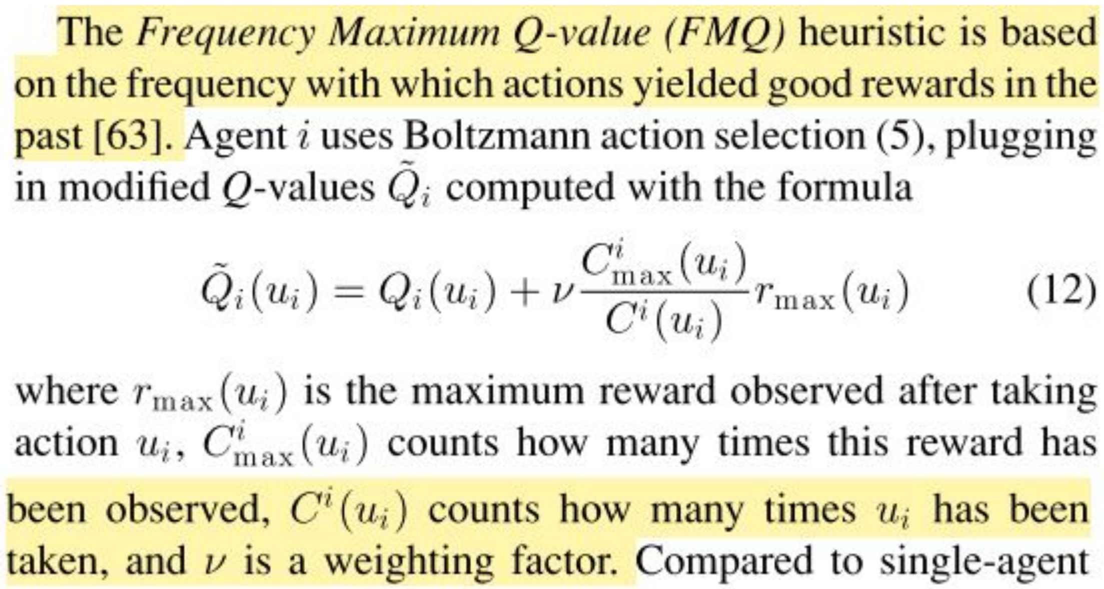


a) Static tasks: Joint Action Learners (JAL) learn joint-action values and employ empirical models of the other agents' strategies [62]. Agent i learns models for all the other agents $j \neq i$, using

$$\hat{\sigma}_j^i(u_j) = \frac{C_j^i(u_j)}{\sum_{\tilde{u}_j \in U_j} C_j^i(\tilde{u}_j)} \tag{11}$$

where $\hat{\sigma}_j^i$ is agent i's model of agent j's strategy and $C_j^i(u_j)$ counts the number of times agent i observed agent j taking action u_j. Several heuristics are proposed to increase the learner's Q-values for the actions with high likelihood of getting good rewards given the models [62].

FMQ

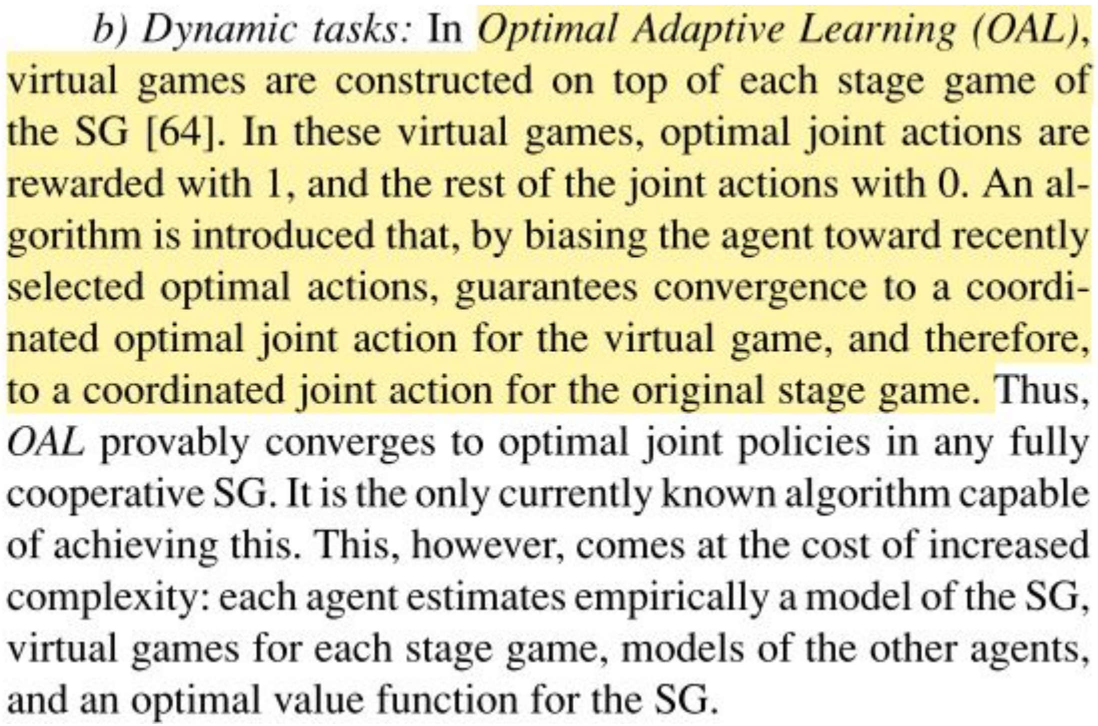


The *Frequency Maximum Q-value (FMQ)* heuristic is based on the frequency with which actions yielded good rewards in the past [63]. Agent i uses Boltzmann action selection (5), plugging in modified Q-values $\tilde{Q}_i$ computed with the formula

$$\tilde{Q}_i(u_i) = Q_i(u_i) + \nu \frac{C^i_{\max}(u_i)}{C^i(u_i)} r_{\max}(u_i) \qquad (12)$$

where $r_{\max}(u_i)$ is the maximum reward observed after taking action u_i, $C^i_{\max}(u_i)$ counts how many times this reward has been observed, $C^i(u_i)$ counts how many times u_i has been taken, and ν is a weighting factor. Compared to single-agent

OAL



b) Dynamic tasks: In *Optimal Adaptive Learning (OAL)*, virtual games are constructed on top of each stage game of the SG [64]. In these virtual games, optimal joint actions are rewarded with 1, and the rest of the joint actions with 0. An algorithm is introduced that, by biasing the agent toward recently selected optimal actions, guarantees convergence to a coordinated optimal joint action for the virtual game, and therefore, to a coordinated joint action for the original stage game. Thus, *OAL* provably converges to optimal joint policies in any fully cooperative SG. It is the only currently known algorithm capable of achieving this. This, however, comes at the cost of increased complexity: each agent estimates empirically a model of the SG, virtual games for each stage game, models of the other agents, and an optimal value function for the SG.

Remarks and Open issues

- Exact measurements of state
    - Communication might be useful
- Exact measurements of actions
    - Communication might be useful
- Curse of dimensionality
    - Not in Distributed Q Learning and FMQ

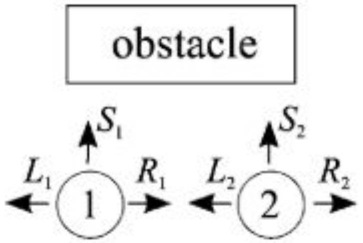
Explicit Coordination Mechanisms

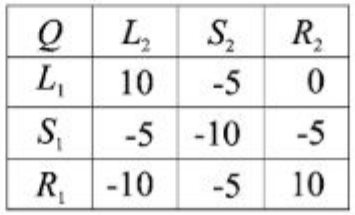
A general approach to solving the coordination problem is to make sure that ties are broken by all agents in the same way.

- Social Conventions
- Roles
- Communication
- Online learning of Coordination

Example 2

If agent 1 < agent 2 -> social convention

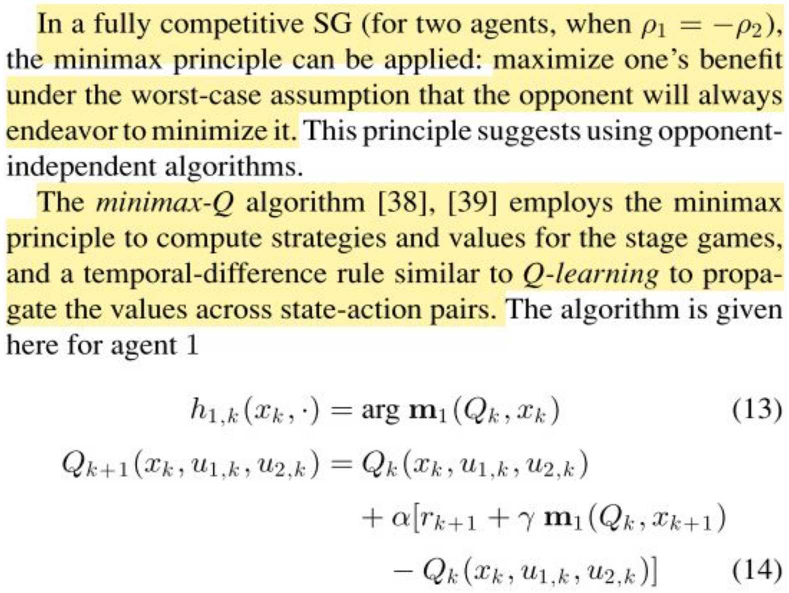




Q	L_2	S_2	R_2
L_1	10	-5	0
S_1	-5	-10	-5
R_1	-10	-5	10

Fully Competitive tasks

Minimax Q learning

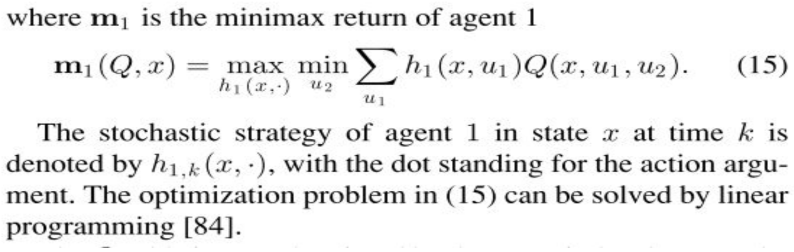


In a fully competitive SG (for two agents, when $\rho_1 = -\rho_2$), the minimax principle can be applied: maximize one's benefit under the worst-case assumption that the opponent will always endeavor to minimize it. This principle suggests using opponent-independent algorithms.

The *minimax-Q* algorithm [38], [39] employs the minimax principle to compute strategies and values for the stage games, and a temporal-difference rule similar to *Q-learning* to propagate the values across state-action pairs. The algorithm is given here for agent 1

$$h_{1,k}(x_k, \cdot) = \arg \mathbf{m}_1(Q_k, x_k) \tag{13}$$

$$\begin{aligned} Q_{k+1}(x_k, u_{1,k}, u_{2,k}) = Q_k(x_k, u_{1,k}, u_{2,k}) \\ + \alpha[r_{k+1} + \gamma\, \mathbf{m}_1(Q_k, x_{k+1}) \\ - Q_k(x_k, u_{1,k}, u_{2,k})] \end{aligned} \tag{14}$$

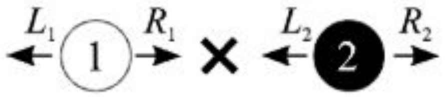


where $\mathbf{m}_1$ is the minimax return of agent 1

$$\mathbf{m}_1(Q, x) = \max_{h_1(x,\cdot)} \min_{u_2} \sum_{u_1} h_1(x, u_1) Q(x, u_1, u_2). \tag{15}$$

The stochastic strategy of agent 1 in state x at time k is denoted by $h_{1,k}(x, \cdot)$, with the dot standing for the action argument. The optimization problem in (15) can be solved by linear programming [84].

Example 3



Q	L_2	R_2
L_1	0	1
R_1	-10	10

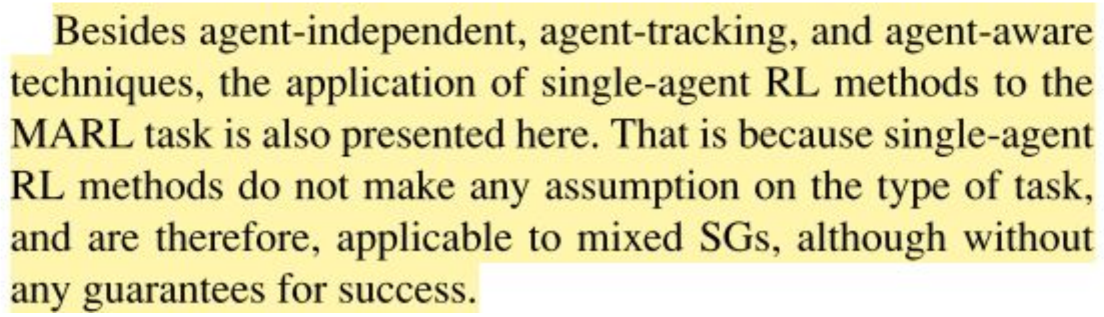
Fig. 4. (Left) An agent (○) attempting to reach a goal (×) while avoiding capture by another agent (●). (Right) The Q-values of agent 1 for the state depicted to the left ($Q_2 = -Q_1$).

The minimax solution for agent 1 in this case is to move left, because for L_1, regardless of what agent 2 is doing, it can expect a return of at least 0, as opposed to -10 for R_1. Indeed, if agent 2 plays well, it will move left to protect the goal. However, it might *not* play well and move right instead. If this is true and agent 1 can find it out (e.g., by learning a model of agent 2), it can take advantage of this knowledge by moving right and achieving the goal.

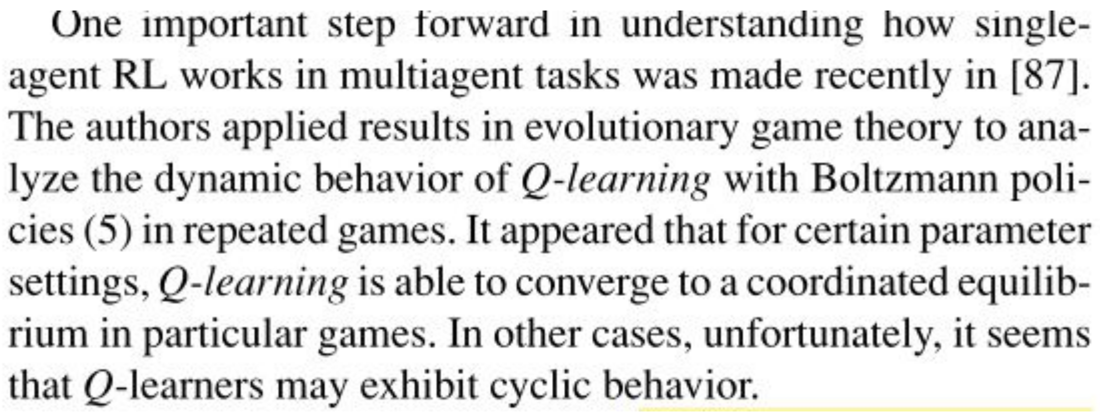
Mixed Tasks

- In mixed SGs, no constraints are imposed on reward functions of agents.
- The influence of game theoretic concepts, like equilibrium concepts, is the strongest in mixed SGs. Also, when multiple equilibria exist in a particular state of an SG, the equilibrium selection problem arises: the agents need to consistently pick their part of the same equilibrium.
- A significant number of algorithms in this category are designed only for static tasks (i.e., repeated general-sum games). In repeated games, one of the essential properties of RL, delayed rewards, is lost.

- Single Agent RL

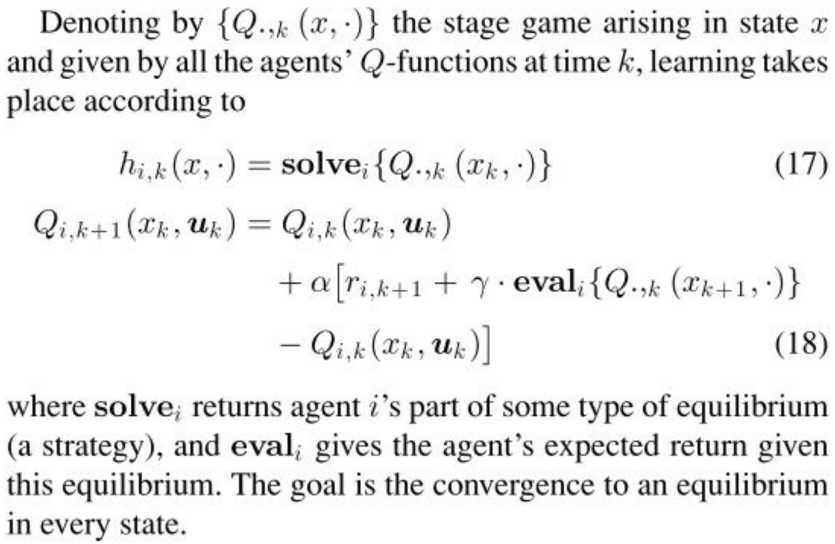


Besides agent-independent, agent-tracking, and agent-aware techniques, the application of single-agent RL methods to the MARL task is also presented here. That is because single-agent RL methods do not make any assumption on the type of task, and are therefore, applicable to mixed SGs, although without any guarantees for success.



One important step forward in understanding how single-agent RL works in multiagent tasks was made recently in [87]. The authors applied results in evolutionary game theory to analyze the dynamic behavior of *Q-learning* with Boltzmann policies (5) in repeated games. It appeared that for certain parameter settings, *Q-learning* is able to converge to a coordinated equilibrium in particular games. In other cases, unfortunately, it seems that *Q*-learners may exhibit cyclic behavior.

- Agent Independent Methods
    - Algorithms that are independent of the other agents share a common structure based on Q-learning, where policies and state values are computed with game-theoretic solvers for the stage games arising in the states of the SG. Solvers can be different from minimax.

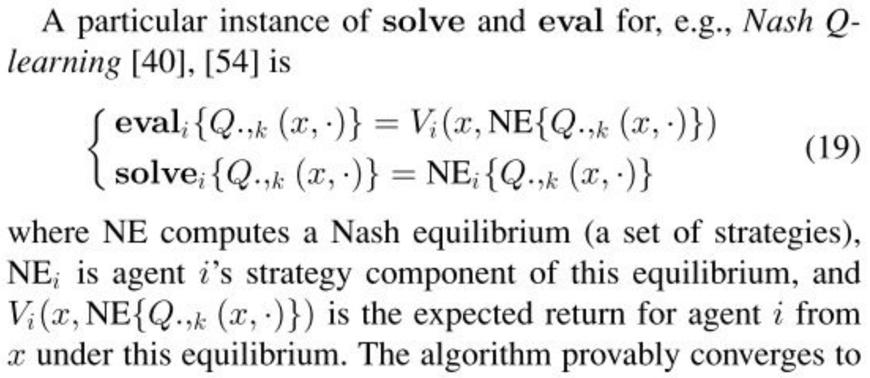


Denoting by $\{Q_{\cdot,k}(x,\cdot)\}$ the stage game arising in state x and given by all the agents' Q-functions at time k, learning takes place according to

$$h_{i,k}(x,\cdot) = \mathbf{solve}_i\{Q_{\cdot,k}(x_k,\cdot)\} \tag{17}$$

$$\begin{aligned} Q_{i,k+1}(x_k,\boldsymbol{u}_k) &= Q_{i,k}(x_k,\boldsymbol{u}_k) \\ &\quad + \alpha\big[r_{i,k+1} + \gamma\cdot\mathbf{eval}_i\{Q_{\cdot,k}(x_{k+1},\cdot)\} \\ &\quad - Q_{i,k}(x_k,\boldsymbol{u}_k)\big] \end{aligned} \tag{18}$$

where $\mathbf{solve}_i$ returns agent i's part of some type of equilibrium (a strategy), and $\mathbf{eval}_i$ gives the agent's expected return given this equilibrium. The goal is the convergence to an equilibrium in every state.

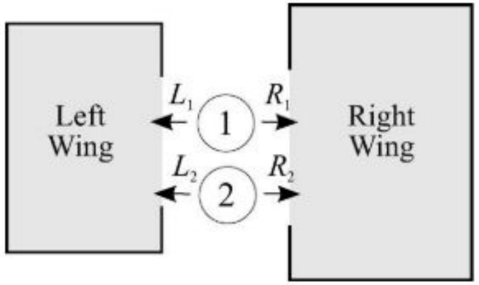


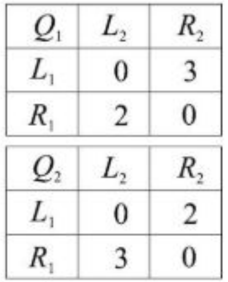
A particular instance of **solve** and **eval** for, e.g., *Nash Q-learning* [40], [54] is

$$\begin{cases} \mathbf{eval}_i\{Q_{\cdot,k}(x,\cdot)\} = V_i(x,\mathrm{NE}\{Q_{\cdot,k}(x,\cdot)\}) \\ \mathbf{solve}_i\{Q_{\cdot,k}(x,\cdot)\} = \mathrm{NE}_i\{Q_{\cdot,k}(x,\cdot)\} \end{cases} \tag{19}$$

where NE computes a Nash equilibrium (a set of strategies), NE_i is agent i's strategy component of this equilibrium, and $V_i(x,\mathrm{NE}\{Q_{\cdot,k}(x,\cdot)\})$ is the expected return for agent i from x under this equilibrium. The algorithm provably converges to

Example 4

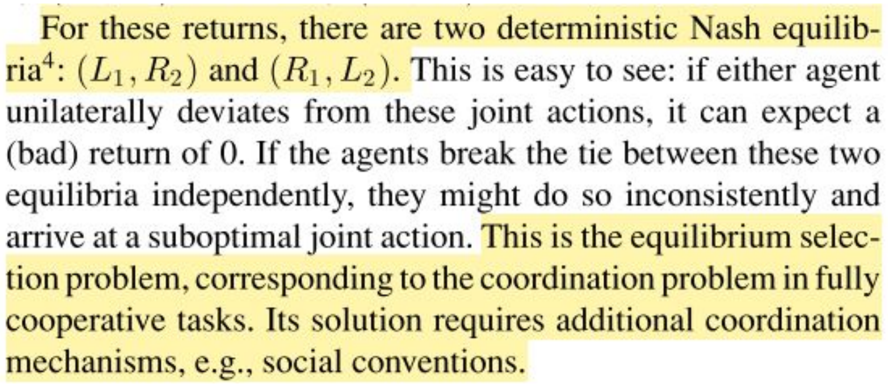




Q_1	L_2	R_2
L_1	0	3
R_1	2	0

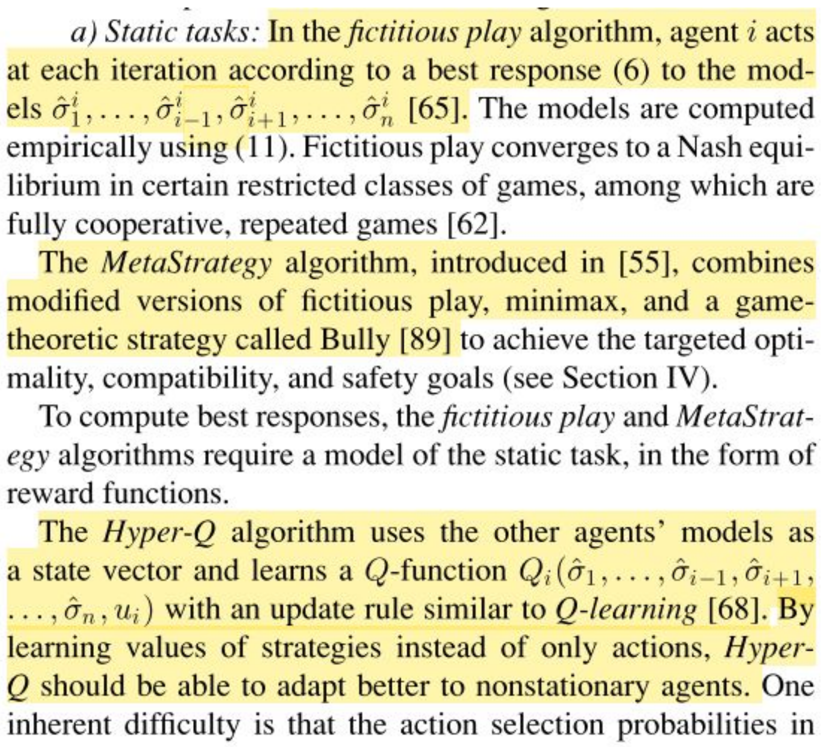
Q_2	L_2	R_2
L_1	0	2
R_1	3	0

Fig. 5. (Left) Two cleaning robots negotiating their assignment to different wings of a building. Both robots prefer to clean the smaller left wing. (Right) The Q-values of the two robots for the state depicted to the left.



For these returns, there are two deterministic Nash equilibria[4]: (L_1, R_2) and (R_1, L_2). This is easy to see: if either agent unilaterally deviates from these joint actions, it can expect a (bad) return of 0. If the agents break the tie between these two equilibria independently, they might do so inconsistently and arrive at a suboptimal joint action. This is the equilibrium selection problem, corresponding to the coordination problem in fully cooperative tasks. Its solution requires additional coordination mechanisms, e.g., social conventions.

- Agent Tracking Methods
  - Models of the other agents' strategies or policies (depending on whether static or dynamic games are considered) are estimated and action taken is using some form of best-response to these models.

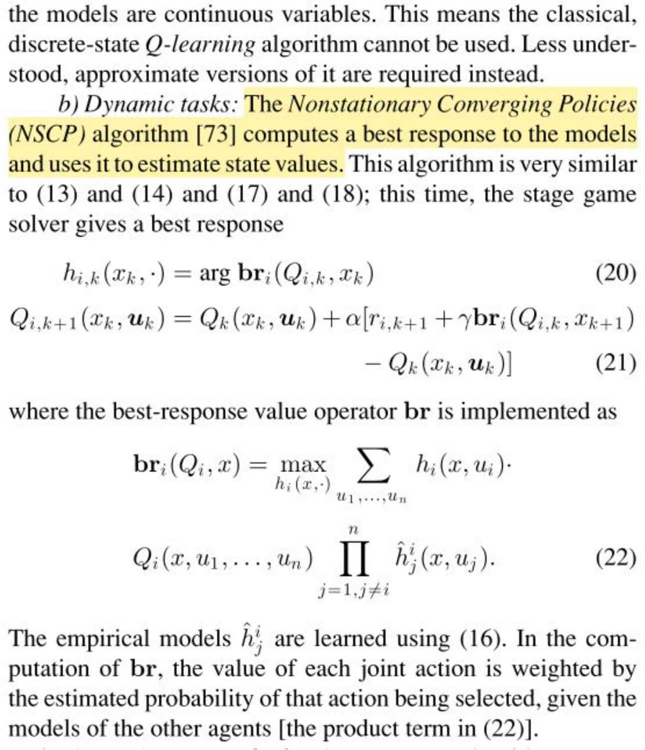


a) Static tasks: In the *fictitious play* algorithm, agent i acts at each iteration according to a best response (6) to the models $\hat{\sigma}_1^i, \ldots, \hat{\sigma}_{i-1}^i, \hat{\sigma}_{i+1}^i, \ldots, \hat{\sigma}_n^i$ [65]. The models are computed empirically using (11). Fictitious play converges to a Nash equilibrium in certain restricted classes of games, among which are fully cooperative, repeated games [62].

The *MetaStrategy* algorithm, introduced in [55], combines modified versions of fictitious play, minimax, and a game-theoretic strategy called Bully [89] to achieve the targeted optimality, compatibility, and safety goals (see Section IV).

To compute best responses, the *fictitious play* and *MetaStrategy* algorithms require a model of the static task, in the form of reward functions.

The *Hyper-Q* algorithm uses the other agents' models as a state vector and learns a Q-function $Q_i(\hat{\sigma}_1, \ldots, \hat{\sigma}_{i-1}, \hat{\sigma}_{i+1}, \ldots, \hat{\sigma}_n, u_i)$ with an update rule similar to *Q-learning* [68]. By learning values of strategies instead of only actions, *Hyper-Q* should be able to adapt better to nonstationary agents. One inherent difficulty is that the action selection probabilities in



the models are continuous variables. This means the classical, discrete-state *Q-learning* algorithm cannot be used. Less understood, approximate versions of it are required instead.

b) Dynamic tasks: The *Nonstationary Converging Policies (NSCP)* algorithm [73] computes a best response to the models and uses it to estimate state values. This algorithm is very similar to (13) and (14) and (17) and (18); this time, the stage game solver gives a best response

$$h_{i,k}(x_k, \cdot) = \arg \mathbf{br}_i(Q_{i,k}, x_k) \tag{20}$$

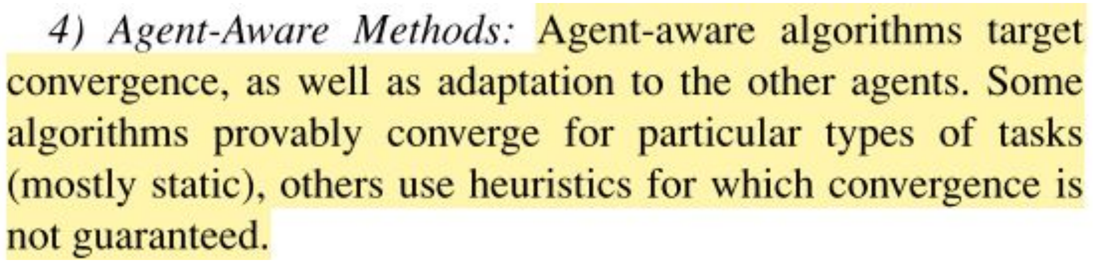
$$Q_{i,k+1}(x_k, \boldsymbol{u}_k) = Q_k(x_k, \boldsymbol{u}_k) + \alpha[r_{i,k+1} + \gamma \mathbf{br}_i(Q_{i,k}, x_{k+1}) - Q_k(x_k, \boldsymbol{u}_k)] \tag{21}$$

where the best-response value operator $\mathbf{br}$ is implemented as

$$\mathbf{br}_i(Q_i, x) = \max_{h_i(x,\cdot)} \sum_{u_1,\ldots,u_n} h_i(x, u_i) \cdot Q_i(x, u_1, \ldots, u_n) \prod_{j=1, j\neq i}^{n} \hat{h}_j^i(x, u_j). \tag{22}$$

The empirical models $\hat{h}_j^i$ are learned using (16). In the computation of $\mathbf{br}$, the value of each joint action is weighted by the estimated probability of that action being selected, given the models of the other agents [the product term in (22)].

- Agent Aware Methods


4) Agent-Aware Methods: Agent-aware algorithms target convergence, as well as adaptation to the other agents. Some algorithms provably converge for particular types of tasks (mostly static), others use heuristics for which convergence is not guaranteed.

  - AWESOME
  - WoLF -IGA
  - WoLF-PHC
  - EXORL

Remarks and Open Issues

- Limited Applications
- Availability of exact task model is assumed
- Game Theory creates a bias towards static (stage-wise) solutions in the dynamic case
- Understanding the conditions under which single agent RL works in mixed SGs

Applications

- Distributed Control
- Robotic Teams
- Automated Trading
- Resource Management

Practical MARL

Scalability (Approximate RL)

Knowledge about task structure and sub-tasks can then be solved (Hierarchical RL)

Incomplete, uncertain state measurements (POMDP)

Conclusions

- Multi-agent Reinforcement Learning is a young, but active and rapidly expanding field of research.

- Significant progress in the field of multi-agent reinforcement learning can be achieved by a more intensive cross-fertilization between the fields of machine learning, game theory and control theory.