Autonomous Agents and Multi-Agent Systems (2020) 34:21
https://doi.org/10.1007/s10458-020-09442-1

The impact of agent definitions and interactions on multiagent learning for coordination in traffic management domains

Jen Jen Chung[1] · Damjan Miklić[2] · Lorenzo Sabattini[3] · Kagan Tumer[4] · Roland Siegwart[1]

Abstract
The state-action space of an individual agent in a multiagent team fundamentally dictates how the individual interacts with the rest of the team. Thus, how an agent is defined in the context of its domain has a significant effect on team performance when learning to coordinate. In this work we explore the trade-offs associated with these design choices, for example, having fewer agents in the team that individually are able to process and act on a wider scope of information about the world versus a larger team of agents where each agent observes and acts in a more local region of the domain. We focus our study on a traffic management domain and highlight the trends in learning performance when applying different agent definitions. In addition, we analyze the impact of agent failure for different agent definitions and investigate the ability of the team to learn new coordination strategies when individual agents become unresponsive.

Keywords Multiagent systems · Cooperation and coordination · Intelligent agents · Agent definitions

✉ Jen Jen Chung
chungj@ethz.ch

Damjan Miklić
damjan@romb-technologies.hr

Lorenzo Sabattini
lorenzo.sabattini@unimore.it

Kagan Tumer
kagan.tumer@oregonstate.edu

Roland Siegwart
rsiegwart@ethz.ch

[1] Eidgenössische Technische Hochschule Zürich, Zurich, Switzerland

[2] RoMb Technologies d.o.o., Zagreb, Croatia

[3] University of Modena and Reggio Emilia, Reggio Emilia, Italy

[4] Oregon State University, Corvallis, USA

Published online: 21 January 2020

1 Introduction

Many traditional multiagent benchmark domains include a strict definition of the agent and the team, examples include the prisoner's dilemma, where each prisoner is an agent; robot soccer, in which each player is an agent; and the predator-prey domain, where each predator or prey are individual agents. However, we are increasingly faced with real world problems where we, as system designers, get to choose the multiagent team structure. For example, in domains such as autonomous traffic management [7, 18], network routing [30] and powerplant control [10]. What constitutes an "agent" in these problems (the controller of a single traffic light or all the traffic lights at an intersection?) is fluid and becomes a design choice rather than a constraint of the domain.

The term *agent factorization* was introduced in [20] and describes the breakdown of a problem into a multiagent system by finding a representation of the full joint state-action space in the union of the individual agent state-action spaces.[1] Yet despite the large body of work in multiagent systems, and especially in multiagent learning for coordination, this concept of designing the agent definitions has not featured much in existing research. Survey papers proposing multiagent system taxonomies [3, 19, 22, 25, 28] tend to focus more on the system architecture, that is, how agents interact and communicate in the domain (if at all). However, these aspects are inherently a function of the agent definition in terms of the (partially observable) Markov decision process that each individual agent is solving.

One of the main challenges of pinning down the contribution of agent definition to the final team performance is the inherent complexity and inter-connectedness of multiagent systems. While there is temptation to simply increase the number of agents in a team without changing the individual agent definition, this does not provide a fair comparison as it fundamentally changes the capabilities of the team, the difficulty of the original problem and so on. Furthermore, the problem domain often does not allow for a straightforward comparison between different agent definitions. For example, in robot soccer, each player can naturally be described as an agent. Other definitions, such as controlling all defenders with a single agent, would require significant reconsideration of how to represent the agent state-actions and interactions, not to mention the practical implications of implementing such an agent.

In this work, we study the impact of agent definition on the performance of multiagent learning for coordination in the warehouse traffic management domain introduced in [7, 12]. This domain is particularly well-suited to our study since the fundamental system dynamics are decoupled from the structure of the multiagent team. Thus, we can independently vary the number of agents in the team and the total information available to the team without changing the difficulty of the underlying problem.

We implement four different agent definitions and also compare against a centralized learning solution. Our results highlight the trade-offs associated with using lower-level definitions, where there are more agents in the team that each have smaller state-action spaces but consequently have a more localized view of the problem, versus higher-level definitions, where there are fewer agents attempting to concurrently learn higher dimensional policies but can each observe a larger portion of the joint space. In essence this provides an insight into the balance between shifting the problem complexity from the learning of

[1] Here we use "agent definition" to avoid confusion with other uses of the term "factorization" in multiagent literature.

coordination (across large numbers of less capable agents), to the learning of individual agent policies, with the centralized agent representing the limit. We also investigate the change in learning performance when more domain information is provided to the agents. Here, our results show that for lower-level agent definitions, the benefits of including additional state information can outweigh the introduced challenges such as increased state dimensionality, especially as the coordination problem becomes more challenging.

This paper is an extended and revised version of our prior work [6], which was presented at the 18th International Conference on Autonomous Agents and Multiagent Systems. The main extensions are additional robustness experiments that examine the effects of various agent failure cases with respect to the different agent definitions. Our analysis explores both the scope of the failure, in terms of the impact within the domain, as well as the direct influence of the agent definition on the multiagent team's ability to discover new coordination strategies after failure events.

The next section provides a summary of the various multiagent taxonomies that have been proposed in the literature and situates the concept of "agent definition" within these categories. A general problem formulation is presented in Sect. 3. In Sect. 4 we introduce the multiagent traffic management domain, which we will use to evaluate the four agent definitions formalized in Sect. 5. The experimental setup is described in Sect. 6 with results and analyses presented in Sect. 7. Finally in Sect. 8 we conclude with a discussion on avenues for future investigation.

2 Background

Several taxonomies have been offered over the past couple of decades to characterize the scope of multiagent problems, specifically multiagent learning problems. As early as [28], researchers in the field recognized the challenge of categorizing the myriad problems that fall under this label, which is due to the many different axes along which multiagent systems can vary. In general, multiagent problems can be grouped according to variations in environment and agent interaction [28], application and architecture [20], hierarchy [25], decentralization and task coupling [22], homogeneity of the agents' goals, actions and domain knowledge [24], as well as by the class of learning algorithms used to solve the problem [3].

One aspect that is often neglected is the choice of team structure for the problem. The main reason for this is that many multiagent learning domains elicit a natural agent definition, to the extent that it becomes part of the definition of the problem rather than a design variable. In many benchmark multiagent domains such as the rover domain [1], the bar problem [29], and robot soccer [16], agents are often defined according to separable physical entities in the team, for example, individual robots in the rover and soccer domains, and individual clients who visit the bar. However, this does not have to be the case. In our introduction, we suggested having a single agent control all defensive robots in the soccer scenario. Another option would be to have separate agents handling moving and kicking for a single robot. Moreover, elements such as agent interaction and learning architecture (e.g. the difference between independent learners and joint action learners [9]) can be thought of as a direct consequence of how an agent is defined. Parunak [20] highlighted this point and also issued a warning:

> When the problem is easily conceived in terms of such naturally-occurring entities, agents can be applied fairly easily. However, factorizations that are suggested by tra-

ditional analysis but do not correspond to naturally occurring entities (such as the hierarchical decomposition of a factory) can lead to very inefficient agent architectures.

A related problem is dealt with in the field of distributed sensing, where one can choose between using a single centralized processing unit or multiple processing agents that collectively infer knowledge about a particular process. In this research area, much of the work concerns the design of data distribution protocols that are fault tolerant and robust to adversarial attacks [21, 26, 27]. Thus, the focus is on how information should be shared at runtime to improve the outcome of collective data processing. Agent definition also impacts the information distribution across a system; however, each definition prescribes what domain information is available to the agent at runtime. Therefore, in this work, we consider how the available information, given the agent definition, affects the ability of individual agents to coordinate as a team.

Recent work by Castellini et al. [4] has explored this problem from the perspective of decomposing the joint action value function in multiagent reinforcement learning. The authors note that for many multiagent coordination tasks, it is often the case that the decision of an agent is only influenced by a small subset of the other agents. Thus, there may be significant computational gains by considering a more appropriate breakdown of these high dimensional multiagent coordination problems. Specifically, the authors studied the representational capacity of different factorizations for several one-shot games, i.e. decompositions of the joint value function into additive local components that each only involve a subset of the agents. They demonstrated that for many problems, relatively accurate representations can be achieved with simple factorizations, but also showed the limitations of this method when dealing with problems that demand tight coordination between the agents.

In this work, we directly study the impact of the individual agent definition on the ability of the team to learn coordinated strategies in the joint state-action space. We consider teams of concurrently learning agents where we can *independently* vary two aspects of the team structure, i.e. the number of team members and the information available to each team member. As summarized by [19], the challenges of concurrent learning fall under two main categories: structural credit assignment and the dynamics of learning. Both of these areas have enjoyed serious attention from the agents community, notably the introduction of individualized reward shaping [1, 11] to address the former, and teammate modeling [23] or turn-taking [5] to tackle the latter. The overall consensus is that without careful team management, both of these challenges become prohibitive with increased team size and problem dimensionality. Thus, agent definition can play a key role in modulating the complexity of the learning problem.

3 Problem formulation

Multiagent learning for coordination with a team of concurrent learners breaks down a global team problem into a set of parallel learning tasks that are coupled according to the global team performance. The global team problem can be defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, where $\mathcal{S}$ is the set of global states, $\mathcal{A}$ is the set of joint actions, $\mathcal{P}^a_{s,s'} = \Pr\left(S^{t+1} = s' | S^t = s, A^t = a\right)$ are the transition probabilities of the system, and $\mathcal{R}^a_s = \mathbb{E}[R(s,a)]$ are the expected rewards. The goal of the team is to collectively execute actions that maximize the expected return, which is often computed as the sum

of discounted future global rewards. However, each agent i only has access to a locally observable state S_i and its own set of actions $\mathcal{A}_i$, which are reduced components of the global state and joint action, respectively. Specifically, for a team of N agents,

$$\mathcal{A} := \mathcal{A}_1 \times \cdots \times \mathcal{A}_N. \tag{1}$$

Note that there is no strict requirement for the same relationship to exist between the individual agent states and the global state. For example, multiple agents may be able to observe the same component of the global state, or the global state may simply not be jointly observable, i.e. the problem is a Dec-POMDP [2]. The individual agent state can be written as,

$$S_i^t = O\left(S^t\right), \tag{2}$$

where S^t is the global state at time t and O is, in general, a non-invertible observation function.

In this work, we explore the effects of varying the state and action definitions of each agent while maintaining the same underlying global team problem. In terms of varying the agent actions, this means that any decomposition of the agent action sets $\left\{\mathcal{A}_i\right\}$ must obey (1). While several valid decompositions may exist for any given problem, a consequence of needing to satisfy (1) is that the larger the dimensionality of each agent's action space, the fewer the number of agents needed to cover the full action space of the global problem. Furthermore, heterogeneous agent definitions can allow team members to have differing degrees of influence over the system. For example, given two agents, i and j, if $|\mathcal{A}_i| \gg |\mathcal{A}_j|$ then in many cases agent i will have greater influence over the team outcome simply because it can act on a greater portion of the joint action space.

Compared to defining valid agent action spaces, greater flexibility is awarded to the definition of the agent state spaces since the only requirement is that one must be able to derive an individual agent's state from the global state. The global state itself is not dependent on the individual agent state representations. Thus, changes to the definition of $\left\{\mathcal{S}_i\right\}$ only impact the dimensionality of each agent's state space and how difficult it is for the team to solve the underlying problem. The former affects the complexity of the individual agent's learning problem [17], while the latter relates to the joint observability of the team as well as the distributed nature of the team information [2]. In the following section, we describe the target domain of our study, after which we formulate several agent state-action definitions that can be applied to this domain in Sect. 5.

4 Traffic management domain

4.1 Domain description

In this work, we study the effect of agent definition in the warehouse traffic management domain introduced in [7, 12]. See Fig. 1 for an illustration of the domain setup. In this domain, M autonomous ground vehicles (AGVs) deliver packages between various locations in a warehouse. The routes in the warehouse are represented as a high level traffic graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each edge $e \in \mathcal{E}$ defines a single direction of travel between two vertices of the graph, i.e. $e = (u, v)$ is the edge from vertex u to vertex v, where $u, v \in \mathcal{V}$. AGVs compute their paths across this graph according to some cost-based planner such as

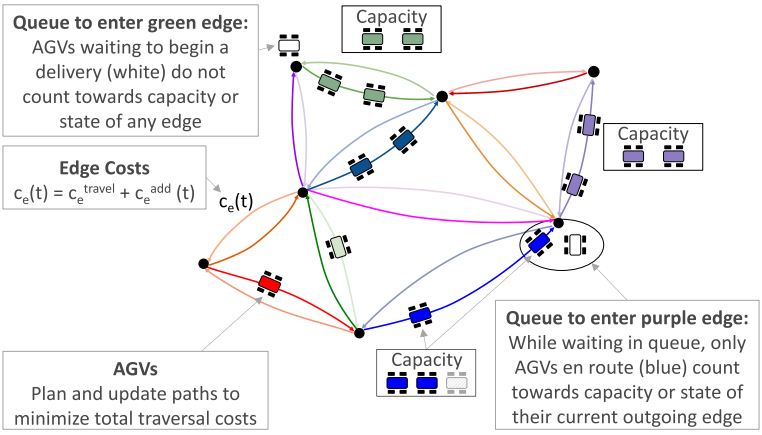


Fig. 1 The environment is described by a traffic graph where each directed edge must obey strict capacity constraints. AGVs cannot enter edges that are already at capacity. Those waiting to transition between edges continue to occupy space on their current edge. The goal of the traffic management domain is to find the set of additional cost functions $c_e^{add}(t)$ that result in the maximal number of successful deliveries

A*. The costs associated with traversing each edge are defined as the sum of a fixed and known cost of travel, c_e^{travel}, and an additional time-varying cost $c_e^{add}(t)$ assigned by the traffic management agents (described in the following subsection),

$$c_e(t) = c_e^{travel} + c_e^{add}(t). \tag{3}$$

AGVs in the system have access to the instantaneous graph costs calculated using Eq. (3), and are able to replan their paths according to the latest costs at any edge transition. However, once they begin traversing an edge, they are committed to continuing along that edge until they reach their next transition. Balking is not permitted in this domain, that is, an AGV cannot spontaneously abandon its current edge and switch to the reverse edge when only part way across. However, an AGV can decide to transition to the reverse edge once it reaches the end of its current edge. This behavior mimics existing warehouse management strategies that enforce lanes of motion and do not allow turning around except at intersections [12]. To capture balking behavior in the current setup, one could include additional transitions at every step between corresponding edges in the current graph to enable an AGV to "turn around" at any point [8]. In the following experiments, all AGVs greedily plan to minimize their traversal costs directly according to the costs at the time of planning.

The domain defines an AGV capacity for each directed edge in the graph cap_e, which imposes a constraint on the motion of the AGVs. The capacity of an edge roughly translates to its available bandwidth and can be determined based on the properties of the underlying physical space represented by that edge (e.g. the size of the free space). During an episode, the number of AGVs on an edge, $n_e(t)$, cannot exceed the capacity of that edge. This means that an AGV planning to transition to an edge which is at capacity must wait on its current edge until there is space. While waiting, it continues to count towards the capacity of its current edge. Thus, without proper management, bottlenecks in the traffic graph can result in cascading congestion throughout the network.

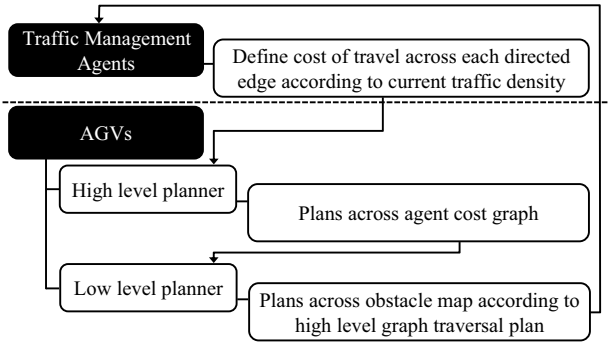


Fig. 2 Hierarchical traffic management formulation, noting the separation between the multiagent traffic management system and the AGVs. The travel space is first decomposed into a high level graph representing the connectivity of different regions in the map. The multiagent system defines the cost of travel across this traffic graph, and the AGVs use these costs to determine their sequence of edge traversals. A lower level planner is then assumed to handle the local collision avoidance procedures through the obstacle map. Figure from [7]

In the original domain, the delivery robots are randomly spawned throughout an episode and are removed from the system once they complete their assigned delivery. Here, we make a slight modification such that all AGVs are initialized at the start of an episode; however, once a delivery is complete, the AGV is not removed, instead it is immediately assigned a new delivery mission which begins at its current location. At this point, if the AGV must wait to enter the first edge on its new path, it does not count towards the capacity of any edge but is considered to be in a "holding zone" until it begins traversal. Thus, the number of AGVs in the system remains constant throughout the episode. This brings the domain closer to the actual dynamics of an automated warehouse [12, 13] and also allows for a more controlled comparison across simulation runs.

4.2 Multiagent traffic management

The task of the traffic management system is to discover the appropriate additional costs, $c_e^{add}(t)$, to apply to each edge in the traffic graph to incentivize the AGVs to avoid congested areas but still reach their destinations in a timely manner. Given a specific sequence of deliveries, an optimal solution to this problem exists; however, the problem quickly becomes intractable for even moderate graph sizes and AGV numbers. The challenge of problem dimensionality also remains if we attempt to jointly learn the costing strategies for all edges. This motivates the use of a multiagent learning framework in which we assign individual agents to manage the traffic in *local* regions of the graph. The problem dimensionality for each agent is therefore much lower compared to the joint learning case. However, the problem complexity now shifts to learning *coordinated* policies across all the agents in the team such that collectively they maximize the global objective. The interaction between the multiagent traffic management team and the AGV traffic is shown in Fig. 2.

Given N agents in a traffic management team, the goal is to concurrently learn the local costing strategies that result in the joint policy $\Pi^* = \{\pi_i\} \forall i \in \{0, \ldots, N-1\}$, which globally produces the highest number of successful deliveries. Agents are defined based on their scope, that is, the component of the global state that they can observe, and the subset of the

joint actions that they can control. In the traffic management domain, this can be represented as the set of edges that an agent manages, as well as the resolution of the information available to each agent regarding the traffic on those edges. For example, an agent may only know the number of AGVs on each edge, or it may also know the exact progress of each AGV along the edge. In general, given the state of each edge to be $\mathbf{s}_e(t)$, then the state for agent i can be defined as,

$$\mathbf{s}_i(t) = \left[\mathbf{s}_e(t)\right], \quad \forall e \in \mathcal{E}_i, \tag{4}$$

where $\mathcal{E}_i$ is the set of edges managed by agent i. Thus, the output actions of agent i are,

$$\mathbf{a}_i(t) = \pi_i\left(\mathbf{s}_i(t)\right) = \left[c_e^{add}(t)\right], \quad \forall e \in \mathcal{E}_i, \tag{5}$$

and the objective of the multiagent team is,

$$\max_{\Pi} G(\Pi) = \textit{total deliveries}, \tag{6}$$

$$\text{s.t.} \quad n_e(t) \leq cap_e \quad \forall t, e \in \mathcal{E}. \tag{7}$$

5 Agent definitions

The traffic management problem provides a domain in which we can investigate the effects of using different multiagent team definitions. The graph structure provides a straightforward decomposition of the joint space whereby subsets of $\mathcal{E}$ are assigned to individual agents to manage. Moreover, practical considerations such as the physical locality of an agent (how far it can sense and communicate) can be naturally incorporated into how the subsets are defined.

In this section, we describe four variants in agent definition which explore two separate axes in multiagent learning. The first axis varies the number of agents in the team while keeping the total information of the system constant. This explores the trade-off between distributing the learning complexity at the agent level (higher dimensional state) versus at the team level (more agents concurrently learning to coordinate). The second axis considers the effect of state resolution to the learning performance in terms of transience and convergence. Intuitively, higher resolutions can discriminate between more system states and provide finer control. However, this comes at the cost of a higher state dimensionality, which can prohibit learning since the space of possible state-actions becomes too large to explore effectively.

5.1 Link agents

The first variant we describe is the agent definition presented in [7]. Here, each *link* agent is assigned to a single directed edge, thus the team consists of $N = |\mathcal{E}|$ agents. The link agent state is simply defined as the total number of AGVs currently traversing the edge, while the link agent output is the additional cost of travel for that edge. That is,

$$s_i^{link}(t) = n_{e_i}(t), \tag{8}$$

$$a_i^{link}(t) = c_{e_i}^{add}(t), \tag{9}$$

where e_i is the edge assigned to link agent i.

Link agents represent the simplest agent definition available to the traffic management domain, since they reduce each agent to a single-input single-output policy. From an individual agent's perspective, this is a straightforward one dimensional learning problem. However, from the team perspective, this represents a very challenging multiagent learning setup. In this case, the structural credit assignment problem is at its most severe [1]; each agent only receives the global team performance as a learning signal, yet its individual contribution to that reward becomes more ambiguous the larger the team size. In addition, the contribution of agent noise is also exacerbated as the number of agents in the team increases [5].

5.2 Intersection agents

The second variant in agent definition that we investigate decomposes the underlying traffic graph according to vertices rather than edges. Specifically, each *intersection* agent is assigned to manage AGV traffic on the set of *incoming* edges of a particular vertex. Thus, the team consists of $N = |\mathcal{V}|$ intersection agents, whose states and actions are,

$$\mathbf{s}_i^{int.}(t) = [n_e(t)], \quad \forall e \in \mathcal{E}_i, \tag{10}$$

$$\mathbf{a}_i^{int.}(t) = [c_e^{add}(t)], \quad \forall e \in \mathcal{E}_i, \tag{11}$$

where $\mathcal{E}_i$ is the set of incoming edges assigned to intersection agent i. Note that the state-action space for each intersection agent in the team can be heterogeneous in this formulation since each agent's dimensionality is defined by the number of incoming edges they manage. That is, the dimensionality of intersection agent i is $|\mathcal{E}_i|$.

Compared to link agents, we generally expect an intersection formulation of a warehouse graph to have fewer agents in the team since for most graphs $|\mathcal{V}| < |\mathcal{E}|$. Thus, the challenges of structural credit assignment and agent noise are reduced when compared to the link agent formulation. However, this comes at the cost of a higher dimensional learning problem for each of the individual agents.

5.3 Incorporating travel time

The final two variants we consider in this study are focused on the effect of state resolution to the multiagent learning problem. As explained in Sect. 3, both the state space definitions and the action space definitions can impact the final team performance. In the link agent and intersection agent formulations described in Sects. 5.1 and 5.2, the state information only consists of the current number of AGVs on the edges. Now we investigate the effect of including additional AGV tracking information.

For each edge, we track $d_e(t)$, the amount of time remaining until the next AGV completes its traversal and will attempt to transition to a new edge or complete its delivery. This value can range from the total time required to traverse an edge (if there are currently no AGVs present) to zero, which represents the case where an AGV has completed its traversal and will transition to a new edge at the next timestep provided it does not violate Eq. (7). This travel time information is incorporated as an additional element in the state vector for each edge. Thus, the travel-time-augmented link agent state becomes,

$$\mathbf{s}_i^{link,time}(t) = \left[n_{e_i}(t), d_{e_i}(t)\right]. \tag{12}$$

Similarly, the augmented intersection agent state is defined as,

$$\mathbf{s}_i^{int.,time}(t) = [n_e(t), d_e(t)], \quad \forall e \in \mathcal{E}_i. \tag{13}$$

Note that the action space for each agent definition remains the same as in Eqs. (9) and (11), respectively.

Augmenting the state with d_e provides a coarse indicator of when the n_e component of the state is about to change. Moreover, it brings the joint state space of the multiagent team closer to full observability of the underlying domain. More information could be provided by individual tracking data on each AGV; however, including such information can be challenging and raises further questions about the design of the agent state definition. For example, d_e only accounts for the progress of the next AGV that will transition off an edge. Instead, one could consider appending the progress of the m-next AGVs to transition to the agent state. This additional tracking information necessarily increases the state dimensionality of the learning agent, yet the fundamental question remains of whether or not this information ultimately helps or hinders the learning of coordinated policies. Further complications arise if attempting to capture the progress of all AGVs since the number of AGVs on each edge will change over time. Thus, if an agent were to directly include the progress of each AGV within its region of control as additional elements in its state input, this would require agent policies that can handle variable-sized input states. While possible through mechanisms such as recurrent or recursive neural networks, we leave this area of investigation to future work.

6 Experimental setup

In the following experiments we use the setup described in [7] and represent the control policy of each agent as a single hidden layer, fully connected neural network.[2] For the link agents (with and without travel time information), we use 16 neurons in the hidden layer, while for the intersection agents we set the number of hidden neurons to four times the number of output neurons. For example, an intersection agent managing AGV traffic on 3 edges will have 12 hidden neurons, its output action dimensionality will be 3, and its input state dimensionality will be 6. Without including travel time information, the intersection agent will only have a state input size of 3. This scaling was chosen as it allows us to maintain comparability between the representational complexity of each of the agent definitions (see the final column of Table 1).

It is worth noting that although we maintain a similar number of total weights in the team, the intersection agent policies must model the correlations between traffic over multiple incoming edges. On the other hand, link agents only observe traffic on a single edge, making an implicit assumption that traffic on different edges are conditionally independent given the global team performance. Thus, intersection agents are tasked with capturing more domain information despite having a similar number of parameters with which to store the encoding. This highlights an important trade-off between having agents with a

[2] We use a logistic activation function at each layer and the final network output is scaled by the base traversal time of the longest edge in the graph.

Table 1 Comparison of agent definitions

Team structure	# Agents	State dim.	Action dim.	Total # weights in team
Link	38	1	1	1254
Link, time	38	2	1	1862
Intersection	11	{2, 3, 4}	{2, 3, 4}	1126
Intersection, time	11	{4, 6, 8}	{2, 3, 4}	1670
Centralized	1	38	38	1254
Centralized, time	1	76	38	1862

more local or a more global view of the multiagent learning task. In this work, we control for the total representational capacity of the whole team; however, future studies may be conducted to investigate other methods for achieving congruence between an agent's scope in the domain and its policy architecture.

To train the weights of the agent policies, we employ cooperative coevolution [14] using the global team performance from Eq. (6) as the fitness evaluation for all policies in the currently tested team. The following subsection provides a brief outline of the cooperative coevolutionary algorithm (CCEA). Note, however, that there is no requirement for using CCEAs to learn coordinated multiagent policies in this domain. Any distributed multiagent learning algorithm, such as multiagent reinforcement learning [19], could be applied in conjunction with any valid control policy representation, provided that the learning algorithm can be trained using coarse reward signals.

6.1 Cooperative Coevolution for Multiagent Learning

Algorithm 1 provides the pseudo-code for our implementation of CCEA. CCEAs evolve multiple populations in parallel; in our case, each agent maintains a population of $K = 10$ neural networks which represents its pool of potential control policies. At the start of evolution, each control policy in the population produces a mutated successor, resulting in a total population of $2K$ neural networks (lines 2–4).

Algorithm 1 Cooperative coevolutionary algorithm

```
1: Initialize N populations of K neural networks
2: for each Population do
3:     Produce K successor solutions
4:     Mutate successor solutions
5: for each Generation do
6:     for k = 1 → 2K do
7:         Randomly sample (without replacement) a policy from each population
8:         Add agent policies to team T_k
9:         G = SimulateTraffic(T_k)                          ▷ Equation (6)
10:        Each agent i ∈ T_k is assigned fitness G
11:    for each Population do
12:        Retain K best networks
13:        Produce K successor solutions
14:        Mutate successor solutions
```

At each generation, a multiagent team is formed by selecting, without replacement, one control policy from each agent's population (lines 7–8). This team is then evaluated by

simulating their performance in the domain (line 9). For our experiments, each episode involves spawning AGVs in the environment that then plan and execute paths based on the output traversal costs from the multiagent traffic management team that is currently being tested. More details on the traffic simulation are provided in the following subsection. At the end of the episode, the performance of the entire team, computed according to Eq. (6), is assigned as the fitness of each of the control policies that made up the team (line 10). Once all $2K$ teams are evaluated, each agent then applies the *selection* step by retaining the K control policies with the best fitness (line 12). In our case, these are the control policies that resulted in the highest number of completed deliveries, G. These K control policies then undergo *mutation* and the process repeats (lines 13–14). For our work, we applied a mutation rate of 10%, that is, at every generation (epoch), each network weight had a 10% probability of undergoing mutation, with added mutation noise drawn from $\mathcal{N}(0, 1)$.

6.2 Warehouse Traffic Graph

We ran our experiments on the simple traffic graph shown in Fig. 3. We tested four instances of the domain that vary the number of AGVs present in the warehouse. Each learning episode contained $\{90, 120, 200, 400\}$ AGVs with half starting at vertex 0 and the other half initialized at vertex 1. Note that the maximum capacity of the traffic graph is 272 AGVs (all edges full). All delivery missions originating at vertex 0 have a goal vertex 1 and vice versa, thus forcing the AGVs to continually move between the two vertices. The layout of the traffic graph is designed such that if all AGVs undertake their A* path computed on the basic traversal costs c^{travel}, then the system will be severely congested along edges $(0, 1)$ and $(1, 0)$. The goal of the multiagent team is to learn the optimal costing strategy that incentivizes detours via vertices 2–10 to enable the highest number of successful deliveries.

We compared the four agent definitions described in Sect. 5 against a centralized traffic management solution where a single agent is tasked to apply costs on all edges of the traffic graph. To maintain comparability, we set the total number of network weights for the centralized agent to be equal to that of the link agent team. Thus, the centralized agent also uses a neural network policy with 16 hidden neurons. Table 1 lists the relevant parameters of the six tested agent team structures for the basic traffic graph, which are evaluated in the following section. Some additional experimental results are provided in "Appendix" that compare the performance of the centralized agent with 16 hidden neurons to its performance when provided with the same relative number of hidden neurons as the intersection agents, i.e. four times the number of edges.

6.3 Agent failure

In addition, we also explore the effect of agent failure for each of the multiagent team structures. Specifically we test two failure sets, the first centered around vertex 6 and the second around vertex 2. Failure is manifested when an agent i no longer applies its learned policy costs, but rather strictly emits the maximum cost, that is,

$$c^{add}_{e_i} = \max_{e \in \mathcal{E}} c^{travel}_{e} \quad \forall e_i \in \mathcal{E}_i, \tag{14}$$

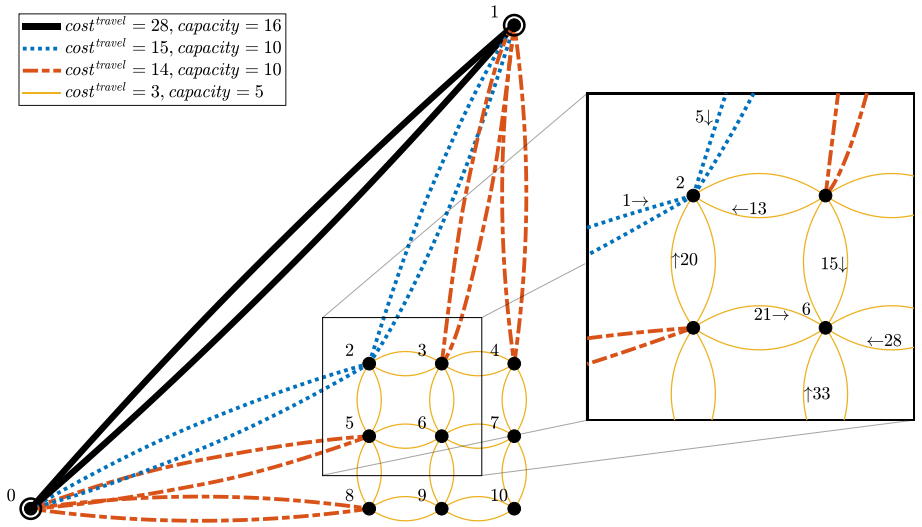


Fig. 3 The basic traffic graph. AGVs are initialized at 0 and 1, traveling to the opposite vertex, i.e. vertex 1 or vertex 0, respectively. Edges $(0, 1)$, and $(1, 0)$ have the highest capacity and represent the shortest path. The goal of the multiagent traffic management team is to incentivize AGVs to take detours to alleviate congestion. The inset shows the link agent indices associated with the incoming edges to vertices 2 and 6. Original underlying figure from [7]

which for our traffic graph in Fig. 3 results in $c_{e_i}^{add} = 28, \forall e_i \in \mathcal{E}_i$. This particular failure scenario is especially severe, since the optimal coordination strategy requires the team to redirect traffic from the main arteries (the direct edges between vertices 0 and 1) through vertices 2 and 6, which means setting the costs for traversing those associated edges to a low value. By assigning fixed high costs, agent failure prevents the use of these edges and forces the discovery of traffic routing strategies through parts of the graph that are initially costlier due to their greater distances.

For each agent definition, we investigated the effect of having a single agent in the team fail. Given the difference in domain influence between a single link agent and a single intersection agent, it is expected that having a single agent fail in either case will result in varying degrees of severity on the team performance. Thus, we also compared the case where all the incoming link agents associated with a vertex fail together, which simulates the same scope of domain failure as the single intersection agent failure case. We expect that a single failed intersection agent will cause a greater initial impact on team performance than a single failed link agent but will be comparable to the case where the set of link agents associated with the same vertex fail. Furthermore, we are particularly interested in studying the recovery behavior for each of the agent definitions. Agents in each team must undergo policy adaptation after the failure event and the transience of this learning process provides some insight into the relationship between the agent definition and the speed and extent to which the team can recover. Table 2 outlines the two sets of experiments that we conducted for this study, the agent indices are labeled on the traffic graph in Fig. 3.

Table 2 Index of agent(s) that fail for each tested set

Team structure	Set 6	Set 2
Link	15	1
Link, time	15	1
Link	{15, 21, 28, 33}	{1, 5, 13, 20}
Link, time	{15, 21, 28, 33}	{1, 5, 13, 20}
Intersection	6	2
Intersection, time	6	2

7 Results and analysis

7.1 Multiagent team performance with increasing congestion

Each of the agent definitions were tested over 30 statistical runs across which the neural network weight initializations were randomized. Each statistical run consisted of 500 epochs (generations), and at each epoch each team was evaluated in an episode 200 timesteps in length. In our experiments we use an initial population size of $K = 10$, which is doubled during the CCEA mutation process (see Algorithm 1). This means that for each agent definition, a single epoch requires 20 domain simulations to test each of the randomly sampled population members. The multiagent team performance for each epoch is then recorded as the highest evaluation achieved over the 20 domain simulations.

Figure 4 shows the progression of team performance across the learning epochs in each of the four domain variants. The plotted values are averaged over the 30 statistical runs with 1σ standard deviations shown by the shaded regions. Figure 5 plots the overall best team performance over the 30 statistical runs and provides an indication of the highest achievable performance throughout the learning procedure.

The first and possibly most noticeable result is that centralized learning (without the inclusion of time information) performs quite well in the mean for lower AGV numbers (90–200 AGVs). It experiences the fastest learning transience and also exhibits a much smaller standard deviation over the mean performance compared to the decentralized, multiagent learning performance. However, the limitations of centralized learning can be seen most clearly in the comparison between Fig. 4c, d (and similarly in Fig. 5). When the underlying problem domain increases in complexity (in our case, higher numbers of AGVs in the warehouse), the performance of the centralized learners saturates while the decentralized learners are able to continue improving. In fact, at 400 AGVs, the best team performance of either of the centralized learners (with and without AGV time) is lower than for the 200 AGV case at {658, 743} deliveries compared to {658, 788}, respectively. The maximum number of deliveries for each agent definition at the end of 500 episodes is shown in Table 3.

The inclusion of AGV travel time into the agent state produces a range of effects depending on the original agent definition. The centralized learner experiences a significant decrease in the mean and maximum learning performance when AGV time is included, while the standard deviation remains roughly around the same order of magnitude. Compare this to the difference between the link and link_t agent definitions. Here we see the opposite relationship whereby the inclusion of AGV travel time produces a substantial improvement in the link_t agent team performance across all warehouse AGV

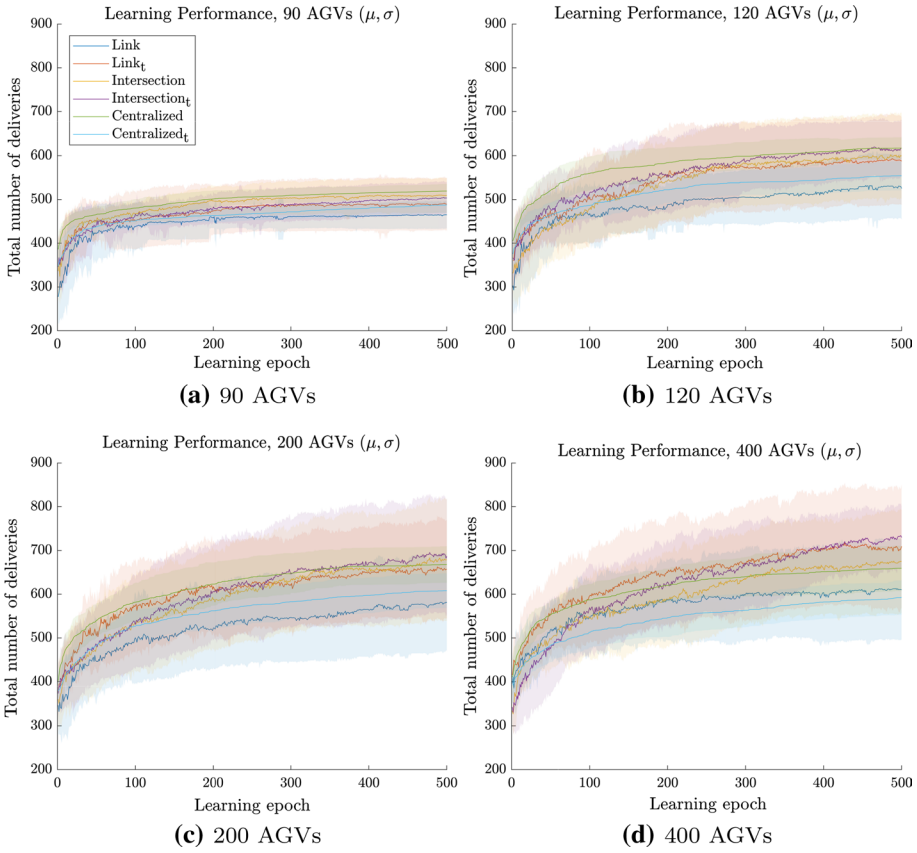


Fig. 4 Average team performance across training epochs. Mean and one standard deviation (from 30 statistical runs) are shown for each set of experiments with increasing numbers of AGVs from (**a**)–(**d**). Best viewed in color (Color figure online)

numbers. In the case of the intersection agent definition, the benefits of including time are less pronounced. A slight trend in improved mean learning performance can be observed as warehouse AGV numbers are increased; however, there is very little difference between the maximum team performances for either intersection agent definition. This suggests that more "local" definitions, i.e. agent state-action spaces that observe a smaller portion of the global state-action space, derive a much greater benefit from expanding their state information and that these benefits can outweigh the challenges they introduce due to increased problem dimensionality. In contrast, including time into the "global" centralized agent state only serves to make the learning problem more difficult without providing any apparent benefits to the overall performance.

Figure 5 shows the maximum achieved performance of any team at each learning epoch. The only case in which a centralized agent performed better than a distributed agent was in the simplest problem domain with only 90 AGVs present in the warehouse. For all other cases, the four distributed agent definitions were able to find better joint policies than the centralized learners, with the gap in improvement widening as the problem complexity rises (i.e. more AGVs to route in the warehouse). However, Fig. 5 shows that distributed

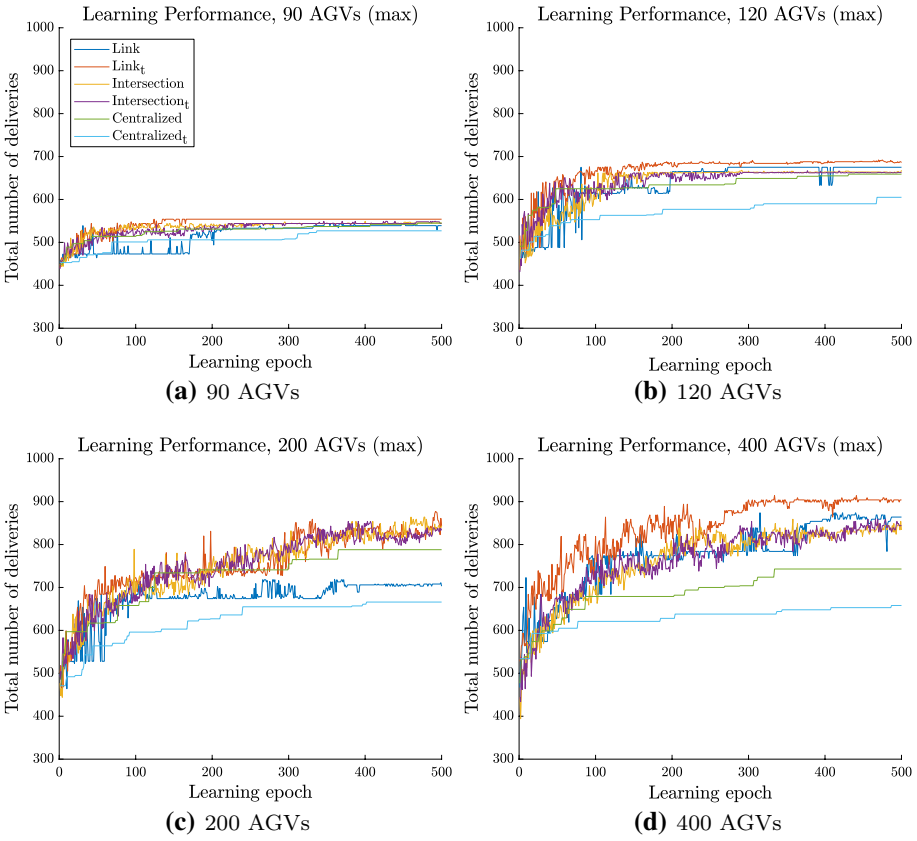


Fig. 5 Best team performance across training epochs. Best viewed in color (Color figure online)

Table 3 Maximum deliveries after 500 learning epochs

Team	90 AGVs	120 AGVs	200 AGVs	400 AGVs
Link	539	675	703	864
Link, time	**554**	**686**	843	**904**
Intersection	545	668	**844**	834
Intersection, time	544	662	837	844
Centralized	545	659	788	743
Centralized, time	527	605	666	658

Values in bold highlight the best performance for the different AGV numbers

learning also displays a high degree of inter-epoch performance variability, and this variability persists for more learning epochs as the problem complexity increases. For example, in the 90 and 120 AGV problems, the maximum learning performance converges after approximately 100–200 epochs; however, for 200 or 400 AGVs its takes well over 300 epochs. This is partially a result of agent noise [5] and is exacerbated by the team randomization which occurs at each epoch. As expected, both of these problems increase in severity the more agents there are in the team. In contrast, the centralized learners exhibit

monotonically increasing performance since evolution will always maintain the best policy from the previous round.

The final result we present focuses on the spread of the team performances at the end of 500 training epochs. The violin plots in Fig. 6 show the distributions over the 30 statistical runs for each of the agent definitions. The four link and intersection result distributions have a much wider spread and are more heavily skewed compared to the centralized learners. Both centralized learners achieved relatively normal distributions with strong agreement between the means and medians, whereas greater disparity can be seen in the corresponding values for the distributed learners. This is another demonstration of the variability in performance that results from agent noise and randomization in the teams during cooperative coevolution. However, for all six agent definitions, the spread in final team performance increased as the underlying problem complexity increased, i.e. more AGVs present in the warehouse.

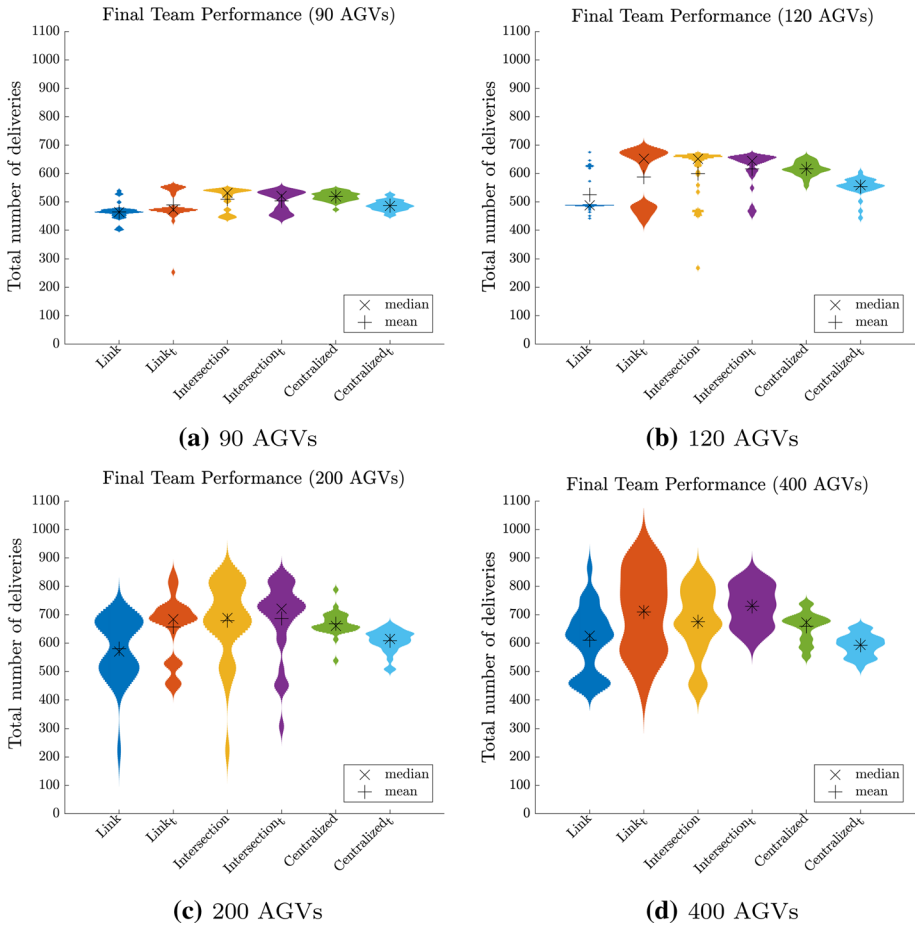
The shape of the final distributions also highlights an interesting relationship between the performances of the two link agent definitions. Both sets of distributions exhibit peaks around the same values, see Fig. 6a–c. However, for the link_t agents, the median performance tends towards the higher valued peaks, whereas the opposite case occurs for the basic link agents. Indeed, each of the peaks in the distributions correspond to local maxima in the joint state-action space. Our results support the notion that the more "local" the agent definition, the more susceptible the team is to getting stuck in local maxima. The AGV travel time information provided sufficient additional domain scope to the link agents to substantially improve their performance, and this improvement is consistent across increasing domain complexity. In contrast, the intersection agent states could already access a wider portion of the joint space, thus the benefits drawn from having additional time information tended to be balanced out by the doubling in state dimensionality.

7.2 Multiagent team performance with agent failure

We investigated the effect of agent failure on team learning performance for the case of 120 AGVs. At this level of congestion, we observed from the previous set of experiments that both vertex 2 and vertex 6 experienced AGV traffic as a result of the final learned policies of the link and intersection agents. However, it is still possible to reroute traffic if a subset of edges associated with either of these vertices becomes blocked (too costly to traverse). The aim of these experiments was to analyze the recovery performance of the team after failure occurs. The following results are collated over 100 statistical runs with failure occurring at epoch 200.

Figure 7 shows the team learning performance for failure set 6. Agent failure at epoch 200 causes all teams to experience a drop in performance, albeit to varying degrees. The failure of link agent 15 causes the weakest drop in team performance, suggesting that up to epoch 200 the agents had not discovered a joint policy that relied heavily on link agent 15's actions. Notably, when the set of link agents $\{15, 21, 28, 33\}$ fail, there is a more pronounced drop and the team is unable to recover to its pre-failure performance within the next 300 learning epochs.

When looking at the mean performance in Fig. 7a, the failure of the link_t agent set causes the greatest impact, while the drop in performance of both intersection agent definitions is of a similar magnitude. Both intersection agent definitions are able to recover much faster in the mean compared to the link_t agents. However, when analyzing the best



(a) 90 AGVs **(b)** 120 AGVs

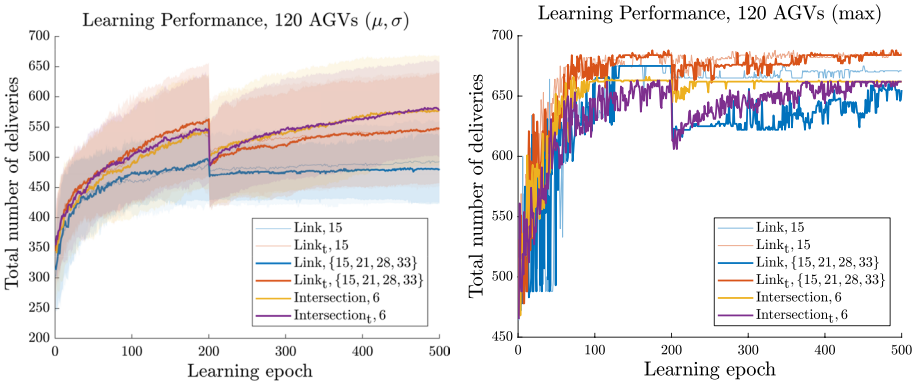
(c) 200 AGVs **(d)** 400 AGVs

Fig. 6 Violin plots of the team performance distributions at the end of 500 training epochs. The '+' symbol represents the mean and the '×' represents the median. All four multiagent team structures produce much wider distributions. Significant skew is observed at lower AGV numbers, whereas at higher AGV numbers the distributions become more symmetrical

team performance over learning epochs (Fig. 7b) we see that intersection and intersection$_t$ agents lag behind the link$_t$ agents both before and after failure occurs.

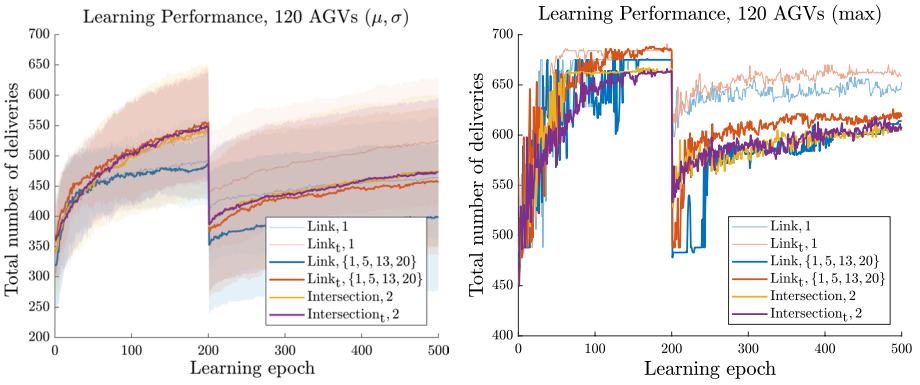
Figure 8 shows the team learning performance for failure set 2. The impact of agent failure is far more pronounced in these plots than for failure set 6. This is because vertex 2 and its associated incoming edges play a more significant role in catering for AGV traffic between vertices 0 and 1. AGV traffic that cannot make it on the direct edge between the start and goal vertices immediately overflow to paths that pass through vertex 2. This is in contrast to vertex 6 which acts as a tertiary route point in the system along with vertices 3 and 5.

In Fig. 8a, b, the mean and maximum team performance drops back to the initial value immediately after edges $\{1, 5, 13, 20\}$ become disabled (intersection/intersection$_t$ agent failure and link/link$_t$ agent set failure). Furthermore, in all cases, teams are unable to recover to their prior performance levels within the subsequent 300 learning epochs. Note



(a) Average team performance **(b)** Best team performance

Fig. 7 Multiagent team performance when agent failure occurs at epoch 200. In this set of experiments we focus on failure at vertex 6. See Table 2 and Fig. 3 for details on the set of failed agents. Best viewed in color (Color figure online)

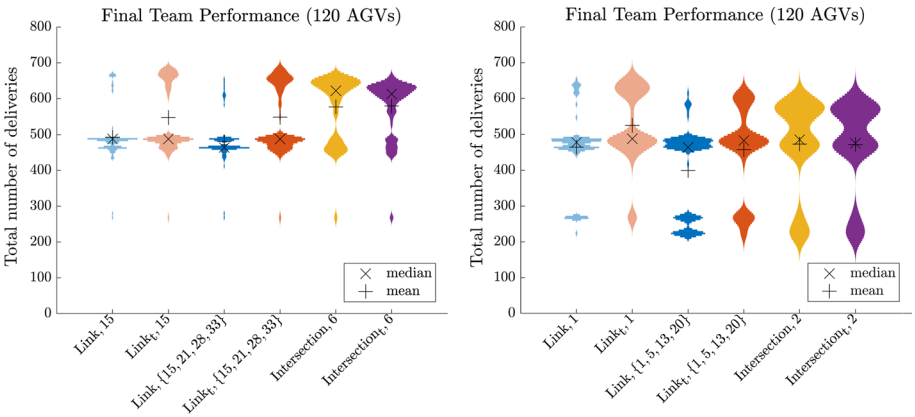


(a) Average team performance **(b)** Best team performance

Fig. 8 Multiagent team performance when agent failure occurs at vertex 2 in epoch 200. Compared to Fig. 7, all teams experience a much more significant drop in performance when failure first occurs. Best viewed in color (Color figure online)

that for this failure set, the teams will be unable to reach their pre-failure optimal levels of coordination since any rerouting around edge 1 or vertex 2 (assuming that the direct edges between vertices 0 and 1 are at capacity) will result in a longer path for the AGVs and so fewer deliveries can be achieved in the same time period. Thus, Fig. 8b shows that even for the case where only link agent 1 fails, the best team performance cannot fully recover for either the link or link$_t$ agent teams.

For the failure cases that involve all incoming edges to vertex 2, the recovery performance is relatively similar between the link$_t$ and both intersection agent definitions. In fact, Fig. 8a shows that on average, these three multiagent team formulations are able to reach or even surpass the performance of the link agent teams where only a single agent (agent 1) has failed. However, analysis of Fig. 8b suggests that the scale of the failure across the domain (the number of edges that become blocked in the graph) is still the main factor that limits the maximal recovery behavior.



(a) Failure at vertex 6 **(b)** Failure at vertex 2

Fig. 9 Comparison of final team performance distributions between different agent definitions and across different failure cases. The distributions show that in general intersection agents are better able to adapt in cases of non-critical agent failure (vertex 6 as opposed to vertex 2) with a substantially higher mean and median in Fig. 9a compared to all other agent definitions. However, for both failure cases, link agents with travel time information were still able to find the best coordinated policy despite performing worse in the mean

As a final comparison, we plot the final team performance distributions in Fig. 9 for the two failure sets. This result highlights the robustness of different agent definitions to different failures across the domain. The plots show three main local maxima for each failure case, each enabling approximately $\{250, 480, 620\}$ AGV deliveries. When compared to Fig. 6b, the main effect of agent failure is to push the mean and median of the team performances toward a lower local maximum. When the failure is less severe, as in the case of failure set 6, intersection agents are able to recover policies that produce coordination solutions close to the higher local maximum, enabling more deliveries. In contrast, link agent teams, with or without time information, are often unable to discover these coordination strategies after failure events. In the case of severe failure, i.e. failure set 2, recovery becomes far more challenging for all agent definitions, with many more teams unable to discover solutions that allow more than 300 deliveries. Nevertheless, comparing the upper regions of the distribution plots shows that, regardless of the failure severity, intersection agents are more capable of discovering effective recovery behaviors compared to link agents. This suggests that teams made up of agents with access to a greater scope of the full domain may be more robust to individual agent failures. Conversely, multiagent systems that exhibit high possibilities for such failure events may benefit from incorporating such agent definition design concepts.

8 Discussion

When formulating a problem as a multiagent coordination task, the individual agent definitions can result in substantial differences in the learned team performance. Therefore, in multiagent domains that do not present a natural agent definition, it is important to consider

these implications when adjusting agent complexity and team coordination complexity. Our study provides some insight into these trade-offs, demonstrating that very "local" definitions are more susceptible to local optima, which can be overcome by providing more state information, despite the overhead of increased state dimensionality. On the other end of the spectrum we show that including the same additional information can hinder learning when a sufficiently "global" view of the problem is already available to the agents. In the latter case, the increased problem dimensionality outweighs the potential benefits of incorporating more information from the joint state.

Our comparison of centralized, intersection and link agent definitions also showed that for simple domains, it may sometimes be preferable to use a centralized learner. The combined benefits of avoiding issues related to agent noise and structural credit assignment reduces the overall variability in learning performance and can provide faster initial learning transience. However, as problem complexity increases, the performance of centralized learning tends to lag whereas distributed learners are able to continue improving the team performance. Our results suggest that substantially longer training times will be necessary for a centralized learner to achieve comparable performance (see also the supplementary results in "Appendix"). The turning point of this trade-off is extremely challenging to pin down. Indeed it remains an open question as to how we can formalize this trade-off for any particular agent definition and any particular problem domain.

Our study into the impact of agent failure also highlighted the differences in recovery response between the various agent definitions. Notably, mean learning performance over multiple statistical runs was often not indicative of the best team performance, with link_t agents often able to find the best joint policy overall despite lagging behind intersection agents in their mean performance. In particular, the distribution over the final team performances also indicated that intersection agents were able to, more frequently, discover team coordination strategies that resulted in high global rewards. Nevertheless, a multiagent team made up of link_t agents was able to recover the best overall joint policy after failure in all of the tested cases.

8.1 Notes for practitioners

As explained in the introduction, the warehouse traffic management domain is ideally suited to studying the effect of agent definition on multiagent learning for coordination since the team formulation is decoupled from the complexity of the underlying problem. In general, we expect our findings to be applicable to other multiagent coordination domains which have similar characteristics. That is, where the agents participate in the domain not as the physical actors in the space, but by controlling the behavior of the physically interacting elements. For example, through a hierarchical command structure such as packet routing in a communication network [30] or as designers of individual sub-components of a mechanism [15]. How the learning trends described in this paper extend more broadly to other classes of multiagent domains is a highly relevant question for further research.

Of course, one must always consider any higher level domain constraints when deciding how an agent can be defined in any particular multiagent domain. For example, to apply a multiagent team in a physical system, the main constraints on the agent definitions will come directly from the system implementation. These include the availability of sensor data, accessibility to control actuation and whether all or any of these data can be communicated across the multiagent system. Returning to the robot soccer example, if the

individual robots cannot communicate with each other or to a base station, then it is impossible to have an agent that controls all of the defensive robots in the team.

8.2 Future work

This study is intended as a starting point for further, more formal investigations into the effect of agent definition. As stated at the start, and demonstrated in our results, the definition of an agent in the context of its domain has a significant effect on team performance when learning to coordinate. So far we have tested a handful of possible agent definitions on a single multiagent domain under different degrees of problem complexity. There remain a number of axes of variation that are yet to be explored. For example, we have not addressed the case where agents in the team have different definitions (i.e. a mix of link and intersection agents). However, the intersection agent definition in this work is an example where team members have different state-action spaces, and consequently have different levels of impact on the joint state-action space. Intersection agents that connect more edges of the graph have greater access to, and impact on, a larger portion of the joint space. Besides agent heterogeneity, other variations such as the presence of hierarchical structures within the multiagent team can provide greater insight into the overall impact of agent definition on multiagent learning for coordination.

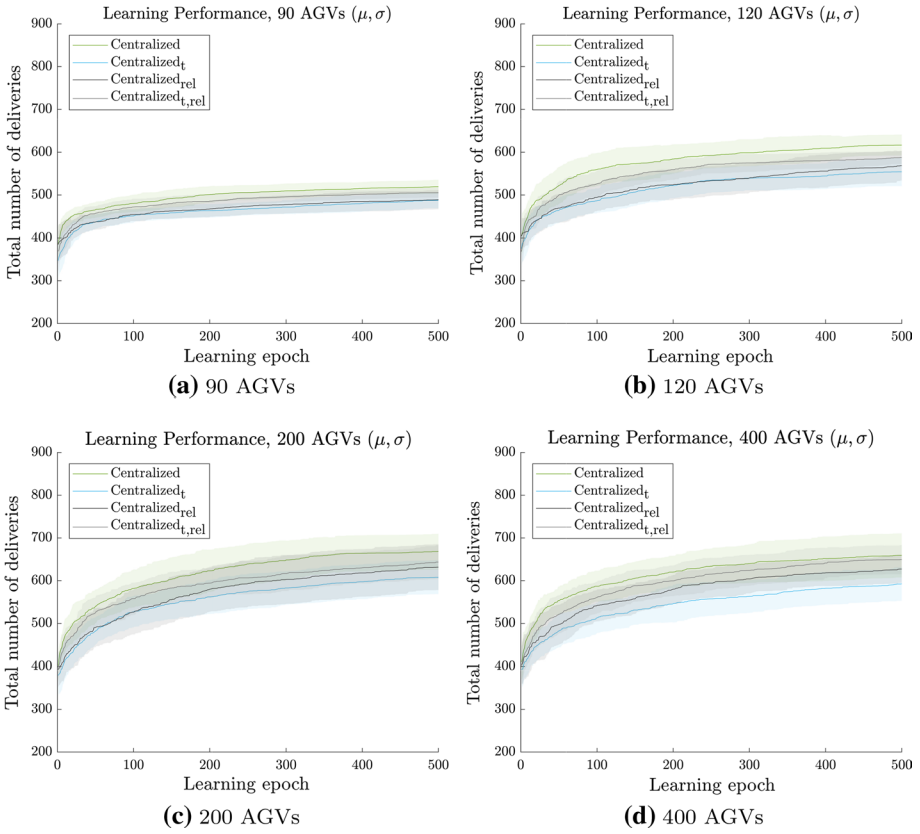
Finally, it is worth noting that in this work we used a relatively simple evolutionary algorithm to optimize for the agent policies. While this was shown to be an adequate training routine for the domain at hand, extending to more complex multiagent coordination problems may require more powerful learning tools. At the same time, using a more powerful learning method may also ameliorate (or exacerbate) some of the learning and coordination challenges highlighted above. Thus, it is certainly worth investigating the degree to which this interaction between training routine and agent definition impacts the overall performance.

Code for the warehouse domain is available open source at https://github.com/JenJenChung/multiagent_learning.

Acknowledgements This paper is an extension of our AAMAS paper [6]. We provide additional experiments analyzing the differences in team performance after agent failure and also expand our discussion with notes for practitioners. This work was partially supported by the EU H2020 project CROWDBOT under Grant Nr. 779942 and by the National Science Foundation under Grant No. IIS-1815886.

Appendix: Additional experiments on the centralized agent network architecture

We conducted additional experiments to assess the performance of the centralized learner when a more complex network architecture was available. In these experiments, the centralized agent uses the same relative number of hidden neurons as the intersection agents, i.e. four times the number of edges, which for 38 edges gives 152 hidden neurons. Thus, the $\text{centralized}_{\text{rel}}$ agent has 11,590 weights defining its control policy while the $\text{centralized}_{\text{t,rel}}$ agent with time information has 17,366 weights. This is over nine times more parameters than the originally tested centralized and $\text{centralized}_{\text{t}}$ agent policies (see Table 1). Results are shown in Figs. 10, 11 and 12 below where the original centralized policies are in green and cyan (no time information and with time information, respectively), while the

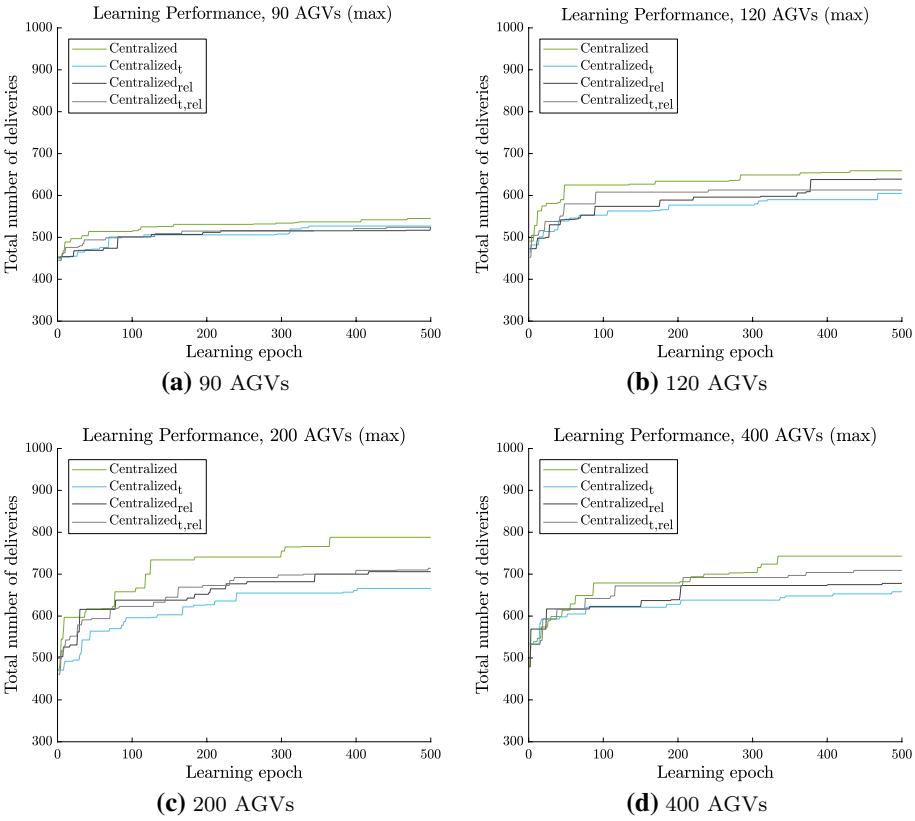


(a) 90 AGVs **(b)** 120 AGVs **(c)** 200 AGVs **(d)** 400 AGVs

Fig. 10 Comparison of the average team performance across training epochs for different centralized agent policy architectures. Green and cyan plots are equivalent to those shown in Figs. 4, 5 and 6, while the grey curves show the performance of centralized agents whose neural network control policies have over nine times more parameters. Mean and one standard deviation (from 30 statistical runs) are shown for each set of experiments with increasing numbers of AGVs from (a)–(d). Best viewed in color (Color figure online)

performance of the larger neural network policies are shown in dark grey and light grey, respectively.

From these results, we see that, in practice, additional policy parameters do not enable substantially better learning performance. The centralized learner with time information is able to improve its learning performance when using the more complex network architecture; however, the centralized learner without time information actually degrades in its performance when more parameters are introduced. Of course, a more comprehensive sweep of possible network architectures would be needed to assess whether there exists a "sweet spot" that balances network representational capacity and learning complexity for each of these cases. Nevertheless, the basic centralized learner with no time information and only 1254 network weights still outperforms all other policy architectures under all computed metrics. A quantitative comparison of the mean and medians of the final distributions is given in Table 4.

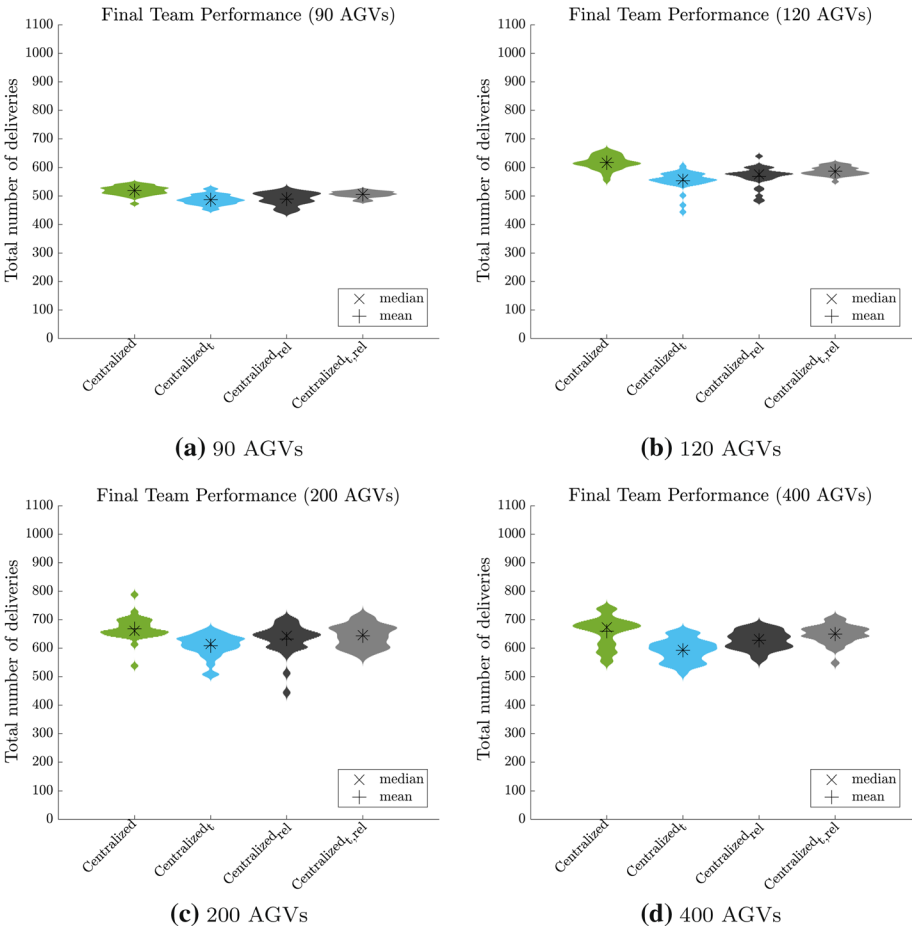


(a) 90 AGVs **(b)** 120 AGVs

(c) 200 AGVs **(d)** 400 AGVs

Fig. 11 Best team performance across training epochs. Best viewed in color (Color figure online)

Finally, we note that in these experiments, we only run our learning algorithm for 500 learning epochs. However, the learning trends suggest that to achieve substantial improvements in the policy performance (e.g. beyond what is currently achieved by the intersection, intersection$_t$ and link$_t$ agents) will require a significantly greater learning effort. The bottleneck may lie in the underlying evolutionary algorithm that is used to train the policies. Thus, future work may consider adapting or replacing this routine with techniques that are tailored to efficiently training neural networks with large numbers of parameters.



(a) 90 AGVs **(b)** 120 AGVs

(c) 200 AGVs **(d)** 400 AGVs

Fig. 12 Violin plots of the team performance distributions at the end of 500 training epochs. The '+' symbol represents the mean and the '×' represents the median. The centralized$_{t,rel}$ agent with 17,366 weights is able to improve on the performance of the centralized$_t$ agent, which only has 1862 weights. However, the basic centralized agent with only 1254 neural network weights produces the highest mean and median performance under all four tested domain complexities

Table 4 Percentage improvement of the basic centralized agent against all other centralized policies

	90 AGVs	120 AGVs	200 AGVs	400 AGVs
Mean (%)				
Centralized$_t$	6.19	10.17	9.00	10.10
Centralized$_{rel}$	5.76	7.79	5.45	4.84
Centralized$_{t,rel}$	2.67	4.72	3.73	1.50
Median (%)				
Centralized$_t$	6.45	9.41	6.96	11.81
Centralized$_{rel}$	5.39	6.49	2.65	6.24
Centralized$_{t,rel}$	2.41	4.87	2.27	3.34

References

1. Agogino, A., & Tumer, K. (2004). Efficient evaluation functions for multi-rover systems. *Genetic and evolutionary computation conference* (pp. 1–11). Seattle, WA: Springer.
2. Bernstein, D. S., Givan, R., Immerman, N., & Zilberstein, S. (2002). The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, *27*(4), 819–840.
3. Busoniu, L., Babuska, R., & De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews*, *8*(2), 156–172.
4. Castellini, J., Oliehoek, F. A., Savani, R., & Whiteson, S. (2019). The representational capacity of action-value networks for multi-agent reinforcement learning. In *Proceedings of the 18th international conference on autonomous agents and multiagent systems* (pp. 1862–1864).
5. Chung, J.J., Chow, S., & Tumer, K. (2018). When less is more: Reducing agent noise with probabilistically learning agents. In *Proceedings of the 17th international conference on autonomous agents and multiagent systems, International foundation for autonomous agents and multiagent systems, Stockholm, Sweden. Extended abstract* (pp. 1900–1902).
6. Chung, J.J., Miklić, D., Sabattini, L., Tumer, K., & Siegwart, R. (2019). The impact of agent definitions and interactions on multiagent learning for coordination. In *Proceedings of the 18th international conference on autonomous agents and multiagent systems*, pp. 1752–1760
7. Chung, J. J., Rebhuhn, C., Yates, C., Hollinger, G. A., & Tumer, K. (2019). A multiagent framework for learning dynamic traffic management strategies. *Autonomous Robots*, *43*(6), 1375–1391.
8. Claes, D., Oliehoek, F., Baier, H., & Tuyls, K. (2017). Decentralised online planning for multi-robot warehouse commissioning. In *Proceedings of the 16th conference on autonomous agents and multiagent systems. International foundation for autonomous agents and multiagent systems* (pp. 492–500).
9. Claus, C., & Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI/IAAI, Madison, WI* (pp. 746–752).
10. Colby, M., Yliniemi, L., Pezzini, P., Tucker, D., Bryden, K.M., & Tumer, K. (2016). Multiobjective neuroevolutionary control for a fuel cell turbine hybrid energy system. In *Proceedings of the genetic and evolutionary computation conference* (pp. 877–884). Denver, CO: ACM
11. Devlin, S., & Kudenko, D. (2011). Theoretical considerations of potential-based reward shaping for multi-agent systems. In *The 10th international conference on autonomous agents and multiagent systems, international foundation for autonomous agents and multiagent systems, Taipei, Taiwan* (Vol. 1, pp. 225–232).
12. Digani, V., Sabattini, L., & Secchi, C. (2016). A probabilistic Eulerian traffic model for the coordination of multiple AGVs in automatic warehouses. *IEEE Robotics and Automation Letters*, *1*(1), 26–32.
13. Digani, V., Sabattini, L., Secchi, C., & Fantuzzi, C. (2015). Ensemble coordination approach in multi-AGV systems applied to industrial warehouses. *IEEE Transactions on Automation Science and Engineering*, *12*(3), 922–934.
14. Ficici, S. G., Melnik, O., & Pollack, J. B. (2005). A game-theoretic and dynamical-systems analysis of selection methods in coevolution. *IEEE Transactions on Evolutionary Computation*, *9*(6), 580–602.
15. Hulse, D., Tumer, K., Hoyle, C., & Tumer, I. (2018). Modeling multidisciplinary design with multiagent learning. In: *Artificial intelligence for engineering design, analysis and manufacturing* (pp. 1–15). FirstView
16. Kitano, H. (1998). *RoboCup-97: Robot soccer world cup I*. New York: Springer.
17. Littman, M.L., Dean, T.L., & Kaelbling, L.P. (1995). On the complexity of solving Markov decision problems. In *Proceedings of the eleventh conference on uncertainty in artificial intelligence* (pp. 394–402).
18. Mannion, P., Duggan, J., & Howley, E. (2016). An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In *Autonomic road transport support systems* (pp. 47–66). New York: Springer.
19. Panait, L., & Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-agent Systems*, *11*(3), 387–434.
20. Parunak, H. V. D. (1996). Applications of distributed artificial intelligence in industry. *Foundations of Distributed Artificial Intelligence*, *2*, 1–18.
21. Rosas, F., Chen, K. C., & Gündüz, D. (2018). Social learning for resilient data fusion against data falsification attacks. *Computational Social Networks*, *5*(1), 10.

22. Sen, S., & Weiss, G. (1999). Learning in multiagent systems. In G. Weiss (Ed.), *Multiagent systems: A modern approach to distributed artificial intelligence* (pp. 259–298). Cambridge, MA: MIT Press.
23. Stone, P., Kaminka, G. A., Kraus, S., & Rosenschein, J. S. (2010). Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *AAAI conference on artificial intelligence, Atlanta, GA* (pp. 1504–1509).
24. Stone, P., & Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, *8*(3), 345–383.
25. Sycara, K. P. (1998). Multiagent systems. *AI Magazine*, *19*(2), 79.
26. Tsitsiklis, J. N. (1993). Decentralized detection. *Advances in Statistical Signal Processing*, *2*(2), 297–344.
27. Veeravalli, V. V., & Varshney, P. K. (2012). Distributed inference in wireless sensor networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *370*(1958), 100–117.
28. Weiß, G. (1996). Adaptation and learning in multi-agent systems: Some remarks and a bibliography. In: G. Weiß & S. Sen (Eds.), *IJCAI'95 workshop on adaption and learning in multi-agent systems* (pp. 1–21). Berlin: Springer.
29. Wolpert, D. H., Wheeler, K. R., & Tumer, K. (1999). General principles of learning-based multi-agent systems. In *Proceedings of the third annual conference on autonomous agents* (pp. 77–83). ACM
30. Ye, D., Zhang, M., & Yang, Y. (2015). A multi-agent framework for packet routing in wireless sensor networks. *Sensors*, *15*(5), 10026–10047.