The Impact of Agent Definitions and Interactions on Multiagent Learning for Coordination

Jen Jen Chung
ETH Zürich
Zürich, Switzerland
jenjen.chung@mavt.ethz.ch

Damjan Miklić
RoMb Technologies d.o.o.
Zagreb, Croatia
damjan@romb-technologies.hr

Lorenzo Sabattini
University of Modena and Reggio Emilia
Reggio Emilia, Italy
lorenzo.sabattini@unimore.it

Kagan Tumer
Oregon State University
Corvallis, Oregon
kagan.tumer@oregonstate.edu

Roland Siegwart
ETH Zürich
Zürich, Switzerland
rsiegwart@ethz.ch

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.

CCS CONCEPTS

• **Computing methodologies** → **Multi-agent systems**; *Cooperation and coordination*; Intelligent agents;

KEYWORDS

AAMAS; ACM proceedings

ACM Reference Format:
Jen Jen Chung, Damjan Miklić, Lorenzo Sabattini, Kagan Tumer, and Roland Siegwart. 2019. The Impact of Agent Definitions and Interactions on Multiagent Learning for Coordination. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

Many traditional multiagent benchmark domains include a strict definition of the agent and the team, e.g. prisoner's dilemma, robot soccer, predator-prey. 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 [4, 12], network routing [20] and powerplant control [6]. What constitutes an "agent" in these problems is fluid and becomes a design choice rather than a constraint of the domain.

The term *agent factorization* was introduced in [14] 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 [2, 13, 15, 18, 19] 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 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, etc., 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 to the performance of multiagent learning for coordination in the warehouse traffic management domain introduced in [4]. 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.

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

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

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 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.

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. In Section 3 we introduce the multiagent traffic management domain, which we will use to evaluate the four agent definitions formalized in Section 4. The experimental setup is described in Section 5 with results and analyses presented in Section 6. Finally in Section 7 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 [19], 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 [19], application and architecture [14], hierarchy [18], decentralization and task coupling [15], homogeneity of the agents' goals, actions and domain knowledge [17], as well as by the class of learning algorithms used to solve the problem [2].

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. However, elements such as agent interaction and learning architecture (e.g. the difference between independent learners and joint action learners [5]) can be thought of as a direct consequence of how an agent is defined. Parunak [14] 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 traditional analysis but do not correspond to naturally occurring entities (such as the hierarchical decomposition of a factory) can lead to very inefficient agent architectures.

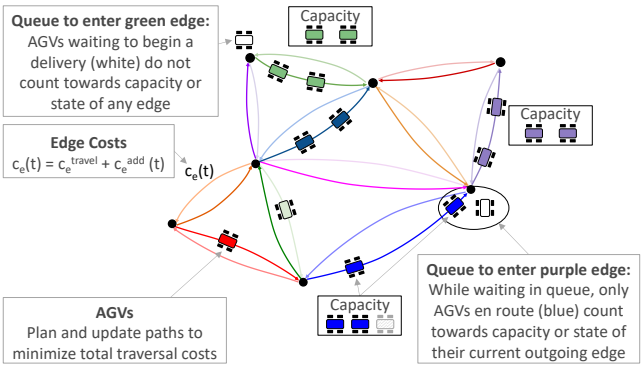


Figure 1: The environment is described by a traffic graph where each directed edge must obey strict capacity constraints. AGVs cannot enter edges that are already at capacity. Those waiting to transition between edges continue to occupy space on their current edge. The goal of the traffic management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

In this work, 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 [13], 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, 7] to address the former, and teammate modeling [16] or turn-taking [3] 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 TRAFFIC MANAGEMENT DOMAIN

3.1 Domain Description

In this work, we study the effect of agent definition in the warehouse traffic management domain introduced in [4]. See Figure 1 for an illustration of the domain set up. 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 A*. The costs associated with traversing each edge are defined as the sum of a fixed and known cost of travel, $cost_e^{travel}$, and an additional time-varying cost $cost_e^{add}(t)$ assigned by the traffic management agents (described in the following subsection),

$$cost_e(t) = cost_e^{travel} + cost_e^{add}(t). \quad (1)$$

AGVs in the system have access to the instantaneous graph costs calculated using Equation (1), 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. 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.

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 [8, 9] and also allows for a more controlled comparison across simulation runs.

3.2 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 Figure 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 $\Pi^* = \{\pi_i\}\ \forall i \in \{0, \cdots, N\text{-}1\}$, which globally produces the highest number of successful deliveries. Agents are defined based on their scope, that is, the component of the joint state that they can observe, and the subset of the joint actions that they can

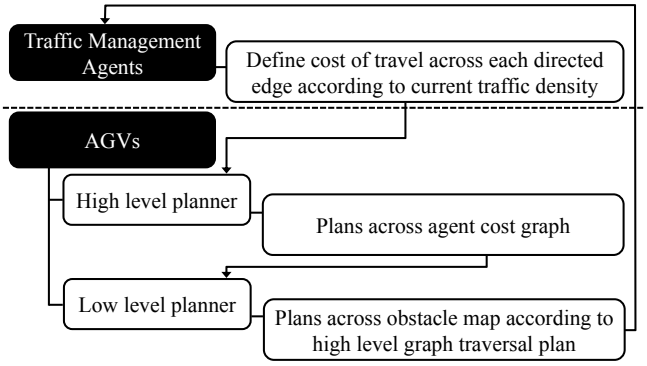


Figure 2: Hierarchical traffic management formulation, noting the separation between the multiagent traffic management system and the AGVs. The travel space is first decomposed into a high level graph representing the connectivity of different regions in the map. The multiagent system defines the cost 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 [4].

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,

$$\mathbf{s}_i(t) = [\mathbf{s}_e(t)], \quad \forall e \in \mathcal{E}_i, \tag{2}$$

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

$$\mathbf{a}_i(t) = \pi_i(\mathbf{s}_i(t)) = \left[cost_e^{add}(t)\right], \quad \forall e \in \mathcal{E}_i, \tag{3}$$

and the objective of the multiagent team is,

$$\max_{\Pi} \quad G(\Pi) = \textit{total deliveries}, \tag{4}$$

$$\text{s.t.} \quad n_e(t) < cap_e \quad \forall t, e \in \mathcal{E}. \tag{5}$$

4 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.

4.1 Link Agents

The first variant we describe is the agent definition presented in [4]. 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,

$$s_i^{link}(t) = n_{e_i}(t), \tag{6}$$

$$a_i^{link}(t) = c_{e_i}^{add}(t), \tag{7}$$

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 [3].

4.2 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,

$$\mathbf{s}_i^{int.}(t) = [n_e(t)], \quad \forall e \in \mathcal{E}_i, \tag{8}$$

$$\mathbf{a}_i^{int.}(t) = [c_e^{add}(t)], \quad \forall e \in \mathcal{E}_i, \tag{9}$$

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.

4.3 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. In the link agent and intersection agent formulations described in Sections 4.1 and 4.2, 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 Equation (5). 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,

$$\mathbf{s}_i^{link,time}(t) = \left[n_{e_i}(t), d_{e_i}(t)\right]. \tag{10}$$

Similarly, the augmented intersection agent state is defined as,

$$\mathbf{s}_i^{int.,time}(t) = [n_e(t), d_e(t)], \quad \forall e \in \mathcal{E}_i. \tag{11}$$

Note that the action space for each agent definition remains the same as in Equation (7) and Equation (9), respectively.

5 EXPERIMENTAL SETUP

In the following experiments we use the setup described in [4] 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 nodes in the hidden layer, while for the intersection agents we set the number of hidden nodes to four times the number of input nodes. For example, an intersection agent managing AGV traffic on 3 edges will have 6 input nodes if it considers AGV travel time information, therefore, its neural network control policy will contain 24 hidden nodes. Without including travel time information, the intersection agent will only have 3 input nodes, and therefore its neural network policy will only have 12 hidden nodes. 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).

To train the weights of the agent policies, we employ cooperative coevolution [10] using the global team performance from Equation (4) 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 [13], could be applied in conjunction with any valid control policy representation, provided that the learning algorithm can be trained using coarse reward signals.

[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.

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** $k = 1 \rightarrow 2K$ **do**
7: Randomly select a policy from each population
8: Add agent policies to team T_k
9: $G = \text{SimulateTraffic}(T_k)$ ▷ Equation (4)
10: Each agent $i \in T_k$ is assigned fitness G
11: **for** each Population **do**
12: Retain K best networks
13: Produce K successor solutions
14: Mutate successor solutions

5.1 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).

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 Equation (4), 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 $\mathcal{N}(0, 1)$.

5.2 Warehouse Traffic Graph

We ran our experiments on the simple traffic graph shown in Figure 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.

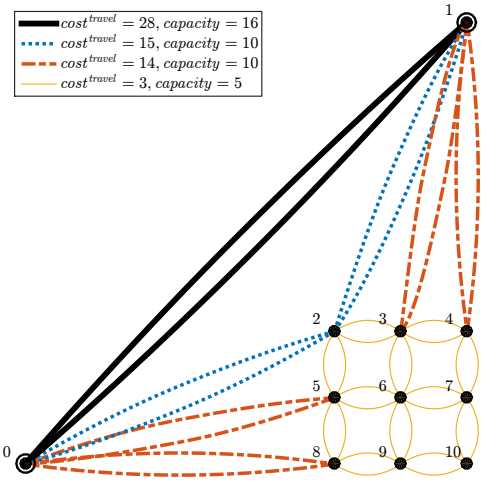


Figure 3: The basic traffic graph. AGVs are initialized at 0 and 1, traveling to the opposite vertex, i.e. vertex 1 or vertex 0, respectively. Edges $(0, 1)$, and $(1, 0)$ have the highest capacity and represent the shortest path. The goal of the multiagent traffic management team is to incentivize AGVs to take detours to alleviate congestion. Figure from [4].

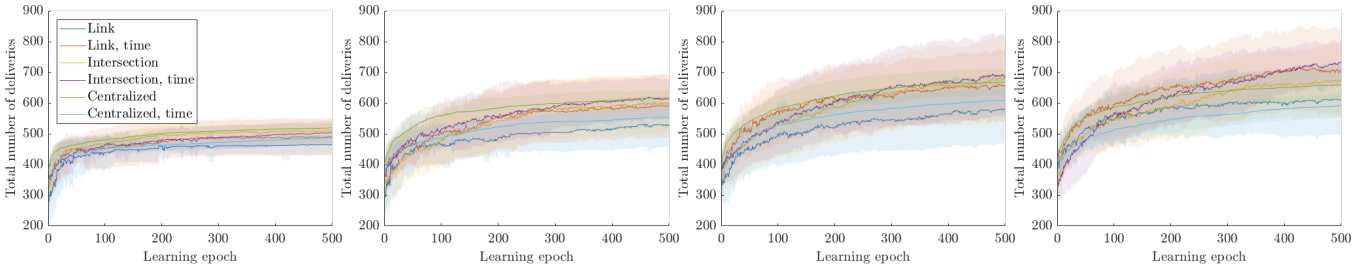
Table 1: Comparison of Agent Definitions

Team structure	# Agents	State dim.	Action dim.	Total # weights in team
link	38	1	1	1254
link, time	38	2	1	1862
int.	11	$\{2, 3, 4\}$	$\{2, 3, 4\}$	1126
int., time	11	$\{4, 6, 8\}$	$\{2, 3, 4\}$	1670
cent.	1	38	38	1254
cent., time	1	76	38	1862

We compared the four agent definitions described in Section 4 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 nodes. Table 1 lists the relevant parameters of the six tested agent team structures for the basic traffic graph.

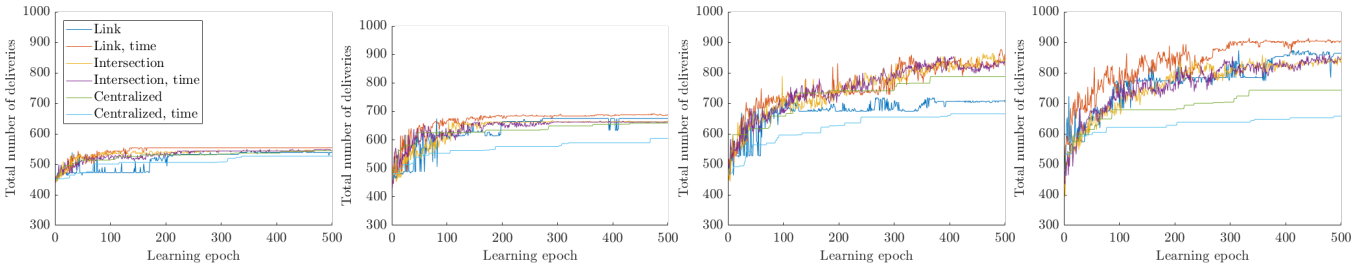
6 RESULTS AND ANALYSIS

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. The average team performances across the epochs, along with 1σ standard deviations, are plotted in Figure 4,



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

Figure 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.



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

Figure 5: Best team performance across training epochs. Best viewed in color.

while Figure 5 shows the best performances across the 30 statistical runs for each agent definition in each of the four domain variants.

These results demonstrate a number of interesting trends. 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 Figures 4c and 4d (and similarly in Figure 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 2.

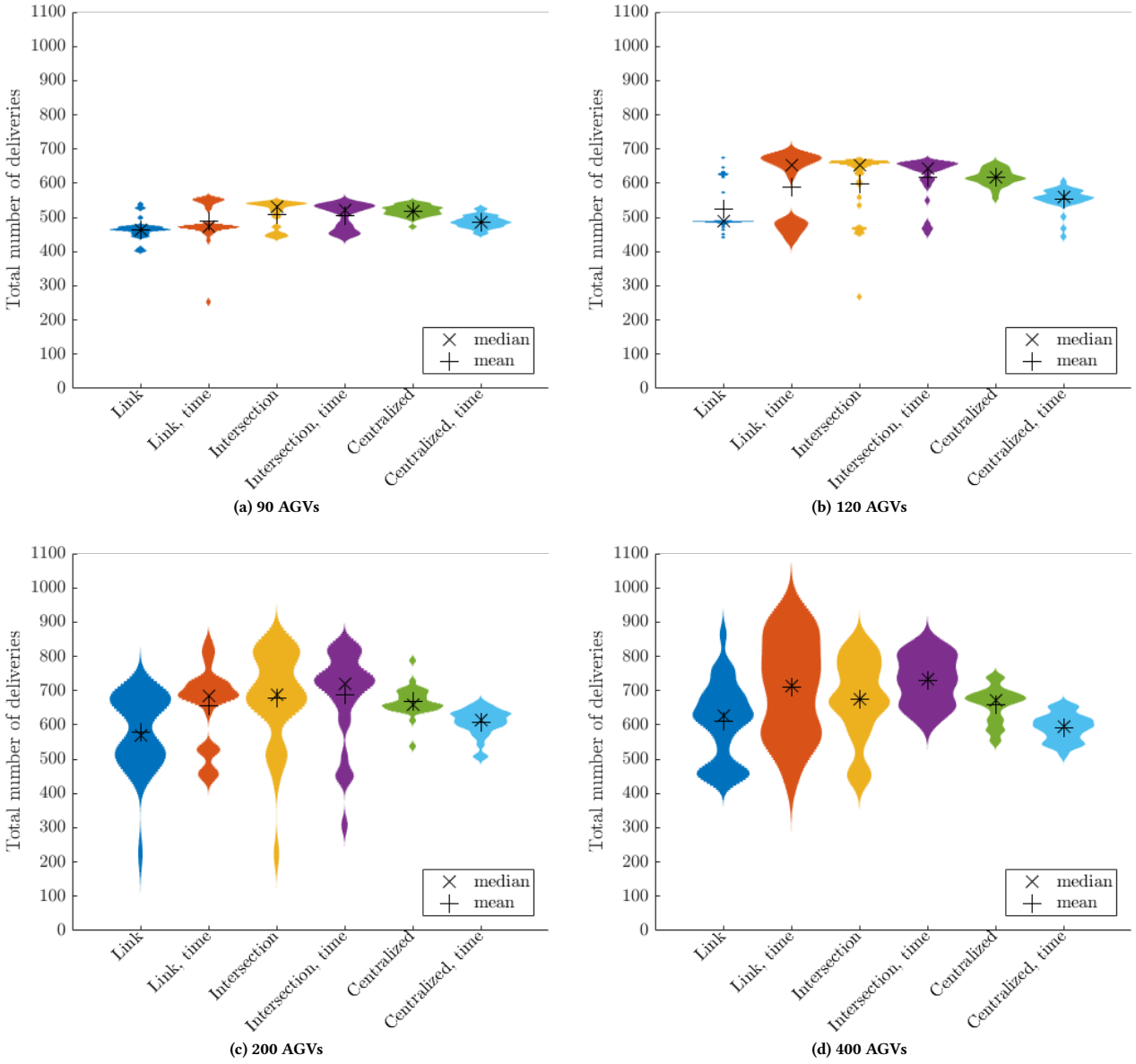
The inclusion of AGV travel time into the agent state produces a range of effects depending on the original agent definition. The centralized learner experiences a significant decrease in the mean and maximum learning performance when AGV time is included, while the standard deviation remains roughly around the same order of magnitude. Compare this to the difference between the link agent definition with and without time information. Here we

Table 2: 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**
int.	545	668	**844**	834
int., time	544	662	837	844
cent.	545	659	788	743
cent., time	527	605	666	658

see the opposite relationship whereby the inclusion of AGV travel time produces a substantial improvement in the link agent team performance across all warehouse AGV 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 derive a much greater benefit from additional 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



(a) 90 AGVs (b) 120 AGVs

(c) 200 AGVs (d) 400 AGVs

Figure 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.

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, Figure 5 shows that distributed 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 [3] 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

Figure 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 Figures 6a, 6b and 6c. However, for the link agents including time information, 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 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 saturate whereas distributed learners are able to continue improving the team performance. 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.

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 [20] or as designers of individual sub-components of a mechanism [11]. 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.

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. These include agent heterogeneity as well as the presence of hierarchical structures within the multiagent team, the former of which we encounter only incidentally in this work. These and other variations can provide greater insight into the overall impact of agent definition on multiagent learning for coordination.

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

ACKNOWLEDGMENTS

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.

REFERENCES

[1] Adrian Agogino and Kagan Tumer. 2004. Efficient evaluation functions for multi-rover systems. In *Genetic and Evolutionary Computation Conference*. Springer, Seattle, WA, 1–11.
[2] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews* 8, 2 (2008), 156–172.
[3] Jen Jen Chung, Scott Chow, and Kagan Tumer. 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, 1900–1902. Extended abstract.
[4] Jen Jen Chung, Carrie Rebhuhn, Connor Yates, Geoffrey A. Hollinger, and Kagan Tumer. 2018. A multiagent framework for learning dynamic traffic management strategies. *Autonomous Robots* (2018), 1–17. https://doi.org/10.1007/s10514-018-9800-z Online first.
[5] Caroline Claus and Craig Boutilier. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI/IAAI*. Madison, WI, 746–752.
[6] Mitchell Colby, Logan Yliniemi, Paolo Pezzini, David Tucker, Kenneth Mark Bryden, and Kagan Tumer. 2016. Multiobjective neuroevolutionary control for a fuel cell turbine hybrid energy system. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, Denver, CO, 877–884.
[7] Sam Devlin and Daniel Kudenko. 2011. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International

Foundation for Autonomous Agents and Multiagent Systems, Taipei, Taiwan, 225–232.
[8] Valerio Digani, Lorenzo Sabattini, and Cristian Secchi. 2016. A Probabilistic Eulerian Traffic Model for the Coordination of Multiple AGVs in Automatic Warehouses. *IEEE Robotics and Automation Letters* 1, 1 (2016), 26–32.
[9] Valerio Digani, Lorenzo Sabattini, Cristian Secchi, and Cesare Fantuzzi. 2015. Ensemble coordination approach in multi-AGV systems applied to industrial warehouses. *IEEE Transactions on Automation Science and Engineering* 12, 3 (2015), 922–934.
[10] Sevan G Ficici, Ofer Melnik, and Jordan B Pollack. 2005. A game-theoretic and dynamical-systems analysis of selection methods in coevolution. *IEEE Transactions on Evolutionary Computation* 9, 6 (2005), 580–602.
[11] Daniel Hulse, Kagan Tumer, Christopher Hoyle, and Irem Tumer. 2018. Modeling multidisciplinary design with multiagent learning. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* (2018), 1–15. FirstView.
[12] Patrick Mannion, Jim Duggan, and Enda Howley. 2016. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In *Autonomic Road Transport Support Systems*. Springer, 47–66.
[13] Liviu Panait and Sean Luke. 2005. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems* 11, 3 (2005), 387–434.
[14] H Van Dyke Parunak. 1996. Applications of distributed artificial intelligence in industry. *Foundations of Distributed Artificial Intelligence* 2 (1996), 1–18.
[15] Sandip Sen and Gerhard Weiss. 1999. Learning in multiagent systems. In *Multiagent systems: A modern approach to distributed artificial intelligence*. MIT Press, Cambridge, MA, 259–298.
[16] Peter Stone, Gal A Kaminka, Sarit Kraus, and Jeffrey S Rosenschein. 2010. Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination. In *AAAI Conference on Artificial Intelligence*. Atlanta, GA, 1504–1509.
[17] Peter Stone and Manuela Veloso. 2000. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots* 8, 3 (2000), 345–383.
[18] Katia P Sycara. 1998. Multiagent systems. *AI magazine* 19, 2 (1998), 79.
[19] Gerhard Weiß. 1996. Adaptation and learning in multi-agent systems: Some remarks and a bibliography. In *IJCAI'95 Workshop on Adaption and Learning in Multi-Agent Systems*, Gerhard Weiß and Sandip Sen (Eds.). Springer Berlin Heidelberg, Montreal, Canada, 1–21.
[20] Dayong Ye, Minjie Zhang, and Yun Yang. 2015. A multi-agent framework for packet routing in wireless sensor networks. *Sensors* 15, 5 (2015), 10026–10047.