Autonomous Robots
https://doi.org/10.1007/s10514-018-9800-z

A multiagent framework for learning dynamic traffic management strategies

Jen Jen Chung[1] · Carrie Rebhuhn[2] · Connor Yates[3] · Geoffrey A. Hollinger[3] · Kagan Tumer[3]

Received: 21 February 2017 / Accepted: 9 August 2018

Abstract
There is strong commercial interest in the use of large scale automated transport robots in industrial settings (e.g. warehouse robots) and we are beginning to see new applications extending these systems into our urban environments in the form of autonomous cars and package delivery drones. This new technology comes with new risks—increasing traffic congestion and concerns over safety; it also comes with new opportunities—massively distributed information and communication systems. In this paper, we present a method that leverages the distributed nature of the autonomous traffic to provide improved traffic throughput while maintaining strict capacity constraints across the network. Our proposed multiagent-based dynamic traffic management strategy borrows concepts from both air traffic control and highway metering lights. We introduce *controller agents* whose actions are to adjust the robots' perceived "costs" of traveling across different parts of the traffic network. This approach allows each robot the flexibility of using its own (potentially proprietary) navigation algorithm, while still being bound by the "rules of the road." The control policies of the agents are defined as neural networks whose weights are learned via cooperative coevolution across the entire traffic management team. Results in a real world road network and a simulated warehouse domain demonstrate that our multiagent traffic management system provides substantial improvements to overall traffic throughput in terms of number of successful trips in a fixed amount of time, as well as faster average traversal times.

Keywords Multiagent systems · Traffic management · Learning for coordination

1 Introduction

Autonomously piloted traffic such as self-driving cars, warehouse robots and package delivery unmanned aerial vehicles (UAVs) have the potential to substantially improve safety, travel times and overall system throughput compared to human-piloted traffic. Such massively distributed autonomous traffic represents the availability of new sensing, communication and data exchange mechanisms compared to existing human-piloted traffic. To fully exploit these capabilities requires a new traffic management paradigm, one where adjustments to local traffic management strategies are aimed directly at achieving globally beneficial traffic behaviors.

A particularly relevant example of automated traffic is robotic package delivery. These services have the potential to revolutionize the way that companies provide delivery services to end users, both in the air and on the ground. However, commercial use of UAVs is predicted to drastically increase the traffic management burden in urban airspaces. NASA NextGen has the goal of enabling safe low-altitude UAV flight within the next 5 years (Kopardekar 2015). Current methods, which rely on human controllers to manage the airspace, become prohibitively costly or unsafe due to constraints on human multitasking and communica-

This work was conducted at Oregon State University and was supported by NASA Grant NNX14AI10G.

✉ Jen Jen Chung
jenjen.chung@mavt.ethz.ch

Carrie Rebhuhn
crebhuhn@mitre.org

Connor Yates
yatesco@oregonstate.edu

Geoffrey A. Hollinger
geoff.hollinger@oregonstate.edu

Kagan Tumer
kagan.tumer@oregonstate.edu

[1] Autonomous Systems Lab, ETH Zürich, Zürich, Switzerland

[2] The MITRE Corporation, McLean, VA, USA

[3] School of Mechanical, Industrial and Manufacturing Engineering, Oregon State University, Corvallis, OR, USA

tion. If we are to allow UAVs and self-driving cars into the airspace and onto our roads on a massive scale, it is clear that an automated approach is necessary. Indeed, whether it is the urban airspace, in our warehouses or on our roads, effective routing strategies will be necessary to manage the large volumes of traffic that want access to these limited spaces.

At the most basic level, traffic management can take the form of metering lights or tollways. Both of these mechanisms serve to influence traffic behavior by enforcing additional time or monetary costs, respectively. Existing implementations of these mostly run on constant time schedules or require direct human intervention to modify, such as in commercial air traffic management where human controllers apportion delays to maintain separation times. However, increasingly we are seeing more adaptive solutions being introduced into real-world domains such as dynamic electronic road pricing (Land Transport Authority of Singapore 2017), autonomous aircraft routing (Prevot et al. 2012; Kopardekar et al. 2016) and, perhaps most commonly, traffic lights that use sensors to determine if cars are waiting to enter an intersection. In their current form, these methods attempt to optimize traffic behavior in only a local region of the full system, and so there remains a significant gap that must be bridged before we can achieve a globally targeted scheme that can keep pace with the highly dynamic demands of dense autonomous traffic.

The key challenge in these types of traffic management problems is to incentivize globally cooperative behaviors from locally greedy entities that are competing for the same resource. That is, we aim to produce smooth and congestion-free travel with a high overall throughput for mobile robots that are operating within the same physical space. In an ideal scenario with a fully cooperative set of traffic, such as in an automated warehouse where all robots act to achieve a common goal, the need for a traffic management system that is robust to a range of traffic profiles as well as potential failures in the system can make a centralized solution intractable. Moreover, the difficulty of this task rises as we relax the requirement for full intra-traffic cooperation and move to domains where we do not assume any organized behavior between individual robots in the system beyond local collision avoidance procedures. For example, in UAV package delivery or autonomous driving domains, the robots in the system are non-cooperative decision makers and must be treated as selfish actors.

The main contribution of our work is to introduce a traffic management layer through which we can apply additional costs to disincentivize robots from planning paths through congested regions of the traffic network (see Fig. 1). By abstracting the traffic network to a graph, we are able to formulate the traffic management system as a distributed multiagent team in which each individual *traffic management agent* observes the traffic along a single directed edge in the graph and adaptively controls the costs applied to robots that traverse that edge. The goal of the multiagent team is for each traffic management agent to learn a custom costing strategy that is applied locally, but can result in improvements in the traffic throughput across the entire system. We show that by integrating a coevolutionary approach into our framework we retain scalability without sacrificing the global view of the overall team goal.

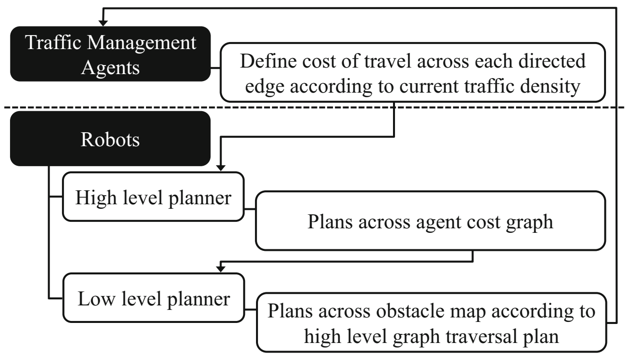

Traffic Management Agents
Define cost of travel across each directed edge according to current traffic density
Robots
High level planner
Plans across agent cost graph
Low level planner
Plans across obstacle map according to high level graph traversal plan

Fig. 1 Hierarchical traffic management formulation, noting the separation between the multiagent traffic management system and the robots. 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 robots 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

We tested our multiagent traffic management framework across a variety of graph structures and a range of traffic profiles. The results show that our proposed system produces smoother overall throughput with shorter wait times compared to existing approaches such as planning purely on the known distances or using fixed costs (e.g. fixed tolls). Results in a real-world traffic domain demonstrated up to 32.5% faster average total travel times which includes 29.0% less overall wait time. Furthermore, we also tested the multiagent system in a high fidelity simulation of a physical traffic environment using the ROS-Gazebo interface to show the validity of those costing strategies that are learned in the high level problem abstraction.

In the following section we highlight existing work in traffic routing and provide background on multiagent learning methods. We present our problem formulation in Sect. 3 and describe our proposed multiagent traffic management framework in Sect. 4. Section 5 provides details on our experimental setup with results presented in Sect. 6. Finally, concluding remarks and suggestions for future research are given in Sect. 7.

2 Related work

2.1 Traffic routing

The problem of mitigating traffic congestion has been tackled by a large body of work in traffic management. Centralized approaches to this problem have considered congestion pricing strategies that aim to strike a balance between the generalized travel cost (a combination of travel time, tolls, etc.) and the number of trips that are executed in the network (Lindsey and Verhoef 2000; Pigou 1920). Many of the dynamic tolling strategies that extend these static tolling methods seek individualized tolls that match the external cost generated by each traveler (i.e. how much they contribute to the delay and congestion in the system). However, these methods rely on having strong models of individual preferences for arrival times and can be relatively complex since they attempt to characterize all the indirect effects of varying the tolls (Arnott and Kraus 1998; Carey and Srinivasan 1993).

Other traffic management approaches shift the focus from optimizing the joint objective over all travelers in the system to simply managing the flow of traffic over the system. For example, methods such as feedback control (Papageorgiou et al. 1997; Stephanedes et al. 1992) or model predictive control (Bellemans et al. 2006; Hegyi et al. 2005a, b) for on-ramp metering manage the rate of traffic entering onto highways to maintain or optimize the traffic flow rate. With the steady move towards intelligent vehicle highway systems, the goal is to distribute intelligence between the roadside infrastructure and the vehicles to allow for a more connected traffic management network (beyond independent ramp metering controllers) while also achieving finer grained control of the overall traffic flow (Baskar et al. 2011).

Existing techniques in automated centralized scheduling and routing have also explored the problem of mitigating traffic congestion. Recent work by Rossi et al. (2018) showed how online fleet rebalancing can be used to mitigate congestion for autonomous mobility on demand services. Qiu et al. (2002) give a comprehensive survey of centralized scheduling methods for automated vehicles. A centralized controller tells every robot in the system when it can travel, and where it can travel. Aubert et al. (2015) proposed a similar framework specifically for UAVs to communicate potential flight plans to a centralized air traffic management system that determines whether to accept or reject the plan. In order to compute a solution, these methods require full state information and are often slow and computationally complex. Dynamic routing methods (Bullo et al. 2011; Taghaboni-Dutta and Tanchoco 1995) manage the time window of each robot through time expanded graphs without needing full state information (Gawrilow et al. 2008). However, these methods become computationally expensive when applied to a fast-changing system.

Other methods for traffic routing have focused on vehicle negotiation to resolve conflicts in congested areas. Large volumes of traffic can use these schemes to resolve local conflicts resulting in improved system performance (Tomlin et al. 1998). For example, peer-to-peer collision avoidance protocols in a software called AgentFly apply an agent-based distributed negotiation approach to the air traffic routing problem (Pechoucek and Sislak 2009). These techniques work well in domains with standardized communication and vehicle abilities. However, robots are diverse in both hardware and software. The method that we propose in this paper allows for these characteristics of the domain and still leaves open the possibility for future progression towards standardization.

Perhaps most similar to our work is the research conducted by Digani et al. (2015, 2016) for managing autonomous ground vehicles (AGVs) in industrial warehouse settings. The authors apply a similar hierarchical approach to their traffic management strategy whereby a roadmap is constructed across the warehouse space, which is then partitioned into sectors. Individual robots use an online planner to determine the sequence of sector traversals to reach their goal and also handle low-level conflicts using negotiation and resource allocation strategies. In our approach, instead of attempting to directly compute the sector traversal costs from the current global state of the traffic system, we allow the multiagent architecture to learn the appropriate costs to apply by observing only local state information and the resultant traffic flow metrics, see Fig. 1.

Unlike previous research, we consider the problem of dynamic traffic routing as a multiagent coordination problem where multiple traffic controllers are simultaneously acting to adjust the flow of traffic in distributed areas of the space. The goal in our work is for this distributed multiagent team to learn an incentivized routing strategy, removing the need for a central controller or vehicle negotiation scheme. Our proposed method does not assign paths to each robot nor does it require the robots to act in a cooperative manner. Our algorithm does not rely on complete knowledge of the individual robot objectives, or high fidelity models of the robot motion; it only assumes that each robot is using a cost-based planner and that the agents are able to manipulate those costs to incentivize the robots to avoid congested areas.

2.2 Multiagent learning approaches

Recent work in multiagent systems has focused on developing effective routing for commercial air traffic across the national airspace. In air traffic, delays and congestion can cascade throughout the system, and are difficult to mitigate and control. Using reinforcement learning agents to manage air traffic through geographical fixes, Agogino and Tumer (2012) were able to reduce airspace congestion by over 90% when compared to current traffic control methods. Cooperative coevolutionary algorithms provided similar demonstrations of the efficacy of a multiagent learning approach (Yliniemi et al. 2014). The setup presented in Agogino and Tumer (2012) has agents control the separation between planes in the airspace. These works provide decentralized control of an airspace, but do not consider movement around obstacles as in our approach.

Our prior work investigated a similar method of using sectors to manage UAV traffic in an urban airspace (Rebhuhn et al. 2015). Instead of applying strict fly/no-fly conditions, the distributed system learned costing strategies to incentivize traffic flow through or around certain sectors of the airspace according to the current traffic profile. The targeted objective was to reduce the overall congestion in the airspace by allowing the agent team to learn policies that accounted for potential cascading effects in the system. This architecture modeled existing solutions in autonomous air traffic control that route airplanes through fixed geographical locations (Agogino and Tumer 2012; Bongiorno et al. 2013; Emami and Derakhshan 2012). Despite showing overall improvements in the number of conflicts experienced across the total airspace, our prior work in Rebhuhn et al. (2015) did not consider physically delaying or separating traffic in the system to prevent over-congestion and so could not provide safety guarantees regarding the density of traffic in any particular region of the airspace. In our current work, we take a safety-critical approach by applying strict capacity constraints and enforce delays on to traffic wanting to enter regions that are currently full. Thus, we shift from optimizing over the number of conflicts in the system to optimizing the system throughput, that is, reducing the total travel time of all traffic in the system.

3 Problem formulation

We consider the problem of routing multiple non-cooperative autonomous robots across a traffic network that can be represented as a complete directed graph $G = (\mathcal{V}, \mathcal{E})$. Each edge $e \in \mathcal{E}$ defines a particular direction of travel between two vertices, i.e. $e = (u, v)$ is the edge from vertex u to vertex v where $u, v \in \mathcal{V}$. The time to traverse each edge, $cost^{travel}$, is fixed and known, however each edge has an additional associated cost, $cost^{add}(t)$, that fluctuates over time t. Individual robots in the system do not have access to the cost functions,

$$cost_e(t) = cost_e^{travel} + cost_e^{add}(t)\,, \tag{1}$$

but do receive the current costs across the graph. We assume that each robot is using a cost-based planner such as A* or D* to compute paths across the graph that will minimize their

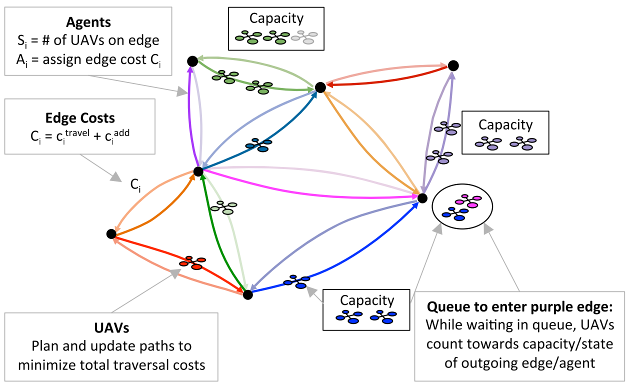

Agents
S_i = # of UAVs on edge
A_i = assign edge cost C_i
Capacity
Edge Costs
C_i = c_i^travel + c_i^add
C_i
Capacity
Capacity
UAVs
Plan and update paths to minimize total traversal costs
Queue to enter purple edge:
While waiting in queue, UAVs count towards capacity/state of outgoing edge/agent

Fig. 2 Each agent in the traffic management system controls the cost of traversing a directed edge across the traffic network graph. The robot traffic (depicted as quadrotor UAVs) form the state of each traffic management agent and must obey strict agent capacity constraints in their motion across the graph

cost of travel given the current set of edge costs. That is, given $cost_e(t)\ \forall e \in \mathcal{E}$, each robot j plans its path according to,

$$\mathcal{P}_j = \underset{\mathcal{P} \subseteq \mathcal{E}}{\arg\min} \sum_{e \in \mathcal{P}} cost_e\,(t), \tag{2}$$

where $\mathcal{P}$ is a path from the robot's current vertex to its goal vertex.

Two additional motion constraints are applied online to the robots as they attempt to traverse the graph. Both of these constraints stem from the realization of a strict capacity limit on each edge. The capacity cap_e represents the maximum number of robots that may be on edge e at any time and can be derived from parameters of the physical traffic space to maintain strict safety and congestion limits. Robots that attempt to enter the traffic network along an edge that is at capacity must wait until another robot has exited that edge before beginning traversal. During this wait period, which is incremented in $wait^{enter}(t)$, the robot does not count towards the capacity of any edge. On the other hand, a robot already on an edge that is attempting to move onto a connecting edge that is at capacity must wait on its current edge until there is space. During this wait period, which is incremented in $wait^{delay}(t)$, the robot counts towards the capacity of its current edge, see Fig. 2. The total amount of time robot j has spent in the system up to time t is computed as,

$$total\_time_j(t) = travel_j(t) + wait_j^{enter}(t) + wait_j^{delay}(t)\,, \tag{3}$$

where $travel_j(t)$ is the amount of time robot j has spent moving along its path. The normalized total travel time of all robots in the system can then be computed as,

$$G(t) = \frac{\sum_{j \in \mathcal{J}(t)} total\_time_j(t)}{\sum_{j' \in \mathcal{J}'(t)} j'}, \tag{4}$$

where $\mathcal{J}(t)$ is the set of all robots that have entered the system up until time t, and $\mathcal{J}'(t) \subseteq \mathcal{J}(t)$ represents the subset of robots that have reached their destination.

Using this construction, we define a multiagent team consisting of $N = |\mathcal{E}|$ agents in which each agent i is assigned to control the advertised cost of travel along a single directed edge $e \in \mathcal{E}$. The state of each agent, $s_i(t)$, is defined as the current number of robots moving along the corresponding edge e summed with the total number of robots waiting to transition off e. The output action of each agent's policy is therefore $\pi_i(s_i(t)) = cost_e^{add}(t)$. The task of the multiagent team is to learn the set of policies, $\Pi^* = \{\pi_0, \ldots, \pi_{N-1}\}$, for which the normalized total travel time at the end of an episode is minimized. That is,

$$\min_{\Pi} \quad G\left(t_{final}\right) \tag{5}$$

$$\text{s.t.} \quad s_i(t) \leq cap_e \quad \forall e \in \mathcal{E}, \forall t \in \left\{0, \ldots, t_{final}\right\} \tag{6}$$

where Eq. (6) enforces the capacity constraints of the graph.

The multiagent policies change the graph costs on which the robots compute their paths and this in turn alters the executed path and therefore the total time a robot spends in the system. In this work we use Eqs. (1) and (2) to compute the graph costs and robot paths; however, our proposed framework only requires the robot paths to be computed as a function of the agent output costs. Thus, other robot planning methods can be substituted provided this assumption is not violated.

It is worth noting that in decomposing the space into a graph we make a number of assumptions about the interactions between the robots and the physical environment. By stating that the time it takes to traverse an edge is fixed and known, we are making two assumptions. First, that the low-level collision avoidance procedures used by the robots when interacting with obstacles and other robots in the environment have a negligible effect on the travel time along an edge. Second, that all paths between two fixed regions of space take the same amount of time to traverse. The traversal of an edge in the graph can be considered equivalent to a trajectory that begins anywhere in one region and crosses a boundary into a second region (each region being represented by a vertex of the graph). This means that the travel time along the path is dependent on the start location in the region and the end location along the boundary, and is therefore not fixed but can be bounded. This problem is exacerbated if multiple homotopy classes of paths exist between the same two regions, e.g. due to obstacles, as the range of paths that are all represented by the same edge in the graph can have very different travel times. The effect of this can be minimized by careful selection of region boundaries for the construction of the traffic graph. In Sect. 6.3 we demonstrate that the agent policies learned on the graph abstraction are robust to variations in the low level plans of the robot traffic by applying the learned traffic management policies to a high fidelity simulation of a physical robot system.

4 Multiagent traffic management

In this work, we approach the traffic management problem by devising a distributed multiagent team in which individual agents are trained to adaptively assign travel costs according to their locally observed traffic. In order to meet global traffic throughput objectives, we use a distributed multiagent learning algorithm such that while agents sense and act locally, they learn to collectively apply a coordinated traffic management strategy.

4.1 Agent definition

Each traffic management agent applies the cost of travel, $cost^{add}$, along a single directed edge of the high level traffic graph. In this way, each agent defines the traversal cost for all robots currently under its jurisdiction (on its directed edge). To compute these costs, we construct each agent as a neural network controller that takes in instantaneous state information and evaluates a corresponding cost to apply to the system. Neural networks are capable of approximating any function to arbitrary accuracy and so have the representational ability to encode continuous state-action control policies while only requiring a coarse representation of the system state (Hornik et al. 1989; Huang et al. 2014). Training a neural network involves setting network weights to encode a control policy that accepts the current state as an input and returns the action that maximizes a particular utility function. In this work, we train the agents with a cooperative coevolutionary algorithm using the normalized travel times computed from Eq. (4) as the fitness evaluation signal. We will discuss coevolution in more detail in the next subsection.

Note that our multiagent traffic management framework does not require agents to be neuro-controllers, nor does it necessitate the use of an evolutionary algorithm to train the team. Any distributed multiagent learning algorithm, such as multiagent reinforcement learning (Panait and Luke 2005), can be applied in conjunction with any valid control policy representation, provided that the learning algorithm can be trained using coarse reward signals. Note that in the traffic management domain, there is no target value from which we can compute an error term and therefore it is difficult to formulate a gradient for optimization purposes. Evolutionary methods are ideally suited to learning in such domains, thus motivating our agent control policy and learning formulation.

In this work, each control policy was defined as a single hidden layer neural network with 1 input, 1 output and 20 hidden nodes, along with sigmoidal activation functions at each layer. We use relatively simple neural network structures to encode our control policies with the aim of demonstrating the ability for a multiagent traffic management team made of simple individual agents to learn complex coordination strategies with only locally available information.

By assigning individual agents to each directed edge, we are able to control the traffic at the resolution of the traffic graph. While this provides us with finer control over the system as compared to, say, having agents control all the outgoing traffic from a single vertex (Rebhuhn et al. 2015), it does raise a challenging multiagent learning problem since the number of agents concurrently learning is now very large. In distributed multiagent learning, each agent must optimize its control policy in a non-stationary environment where at each learning epoch, all agents concurrently update their policies, thereby continually shifting the goal posts for learning coordinated strategies. At the same time, within each epoch, all agents receive the same global reward signal, which does not disambiguate between the individual contributions of each agent. Thus, agents that may be performing well, but are teamed up with poor collaborators will receive a low reward, while the converse is also true. In this paper, we show that despite these challenges, our proposed multiagent traffic management framework is capable of learning distributed control policies that improve the overall traffic throughput. In future work we will investigate how reward shaping methods (Agogino and Turner 2004) can be used to address some of these issues to further improve upon our current work.

4.2 Cooperative coevolution

Cooperative coevolutionary algorithms (CCEAs) are an extension of EAs and have been shown to perform well in cooperative multiagent domains (Ficici et al. 2005). 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 represent 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, this involves stochastically spawning robots 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 simulation, the performance of the entire team, computed according to Eq. (4), is then 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 minimal normalized robot travel time, 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, each network weight had a 10% probability of undergoing mutation, with mutation noise drawn from $\mathcal{N}(0, 1)$.

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 $i = 1 \rightarrow 2k$ do
7:         Randomly select one agent from each population
8:         Add agents to team $T_i$
9:         $G$ = SimulateTraffic ($T_i$)                ▷ Equation (4)
10:        Each agent $j \in T_i$ is assigned fitness $G$
11:    for each Population do
12:        Retain $k$ best networks
13:        Produce $k$ successor solutions
14:        Mutate successor solutions
```

4.3 Robot traffic simulation

We train and test each multiagent traffic management team in a simulated traffic scenario described in Algorithm 2. At each timestep of each simulation round, up to max^{rob} new robots are randomly spawned at each source vertex with a given probability p^{rate} (lines 5–7). Agents then compute their local state and associated cost action according to the current distribution of robots in the traffic graph (lines 8–10). These costs are added to the fixed travel costs along each edge of the graph to form the final graph costs over which the robots plan their paths (lines 11–12). Robots that have completed the traversal of their current edge are able to replan their path based on the latest graph costs (lines 14–15). All robots then attempt to move along their path but must obey edge capacity constraints when attempting to transition to a new edge, i.e. if the edge they want to transition to is at capacity, they must wait and continue to occupy a position on their current edge. Newly spawned robots that cannot enter an edge, must also wait, but do not count towards the capacity of any edge (line 16). Robots that reach their goal are removed from the system (line 17). Finally, the traffic metrics are logged and used to compute the global system performance of the tested multiagent traffic management team (lines 18–19).

Algorithm 2 SIMULATETRAFFIC(T_i)

```
1:  Initialize traffic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
2:  Initialize probabilistic traffic generation parameters for each $v \in \mathcal{V}$:
    $max^{rob}$, $p^{rate}$
3:  Initialize robot traffic $\mathcal{M} \leftarrow \emptyset$
4:  for $t = 0 \rightarrow t_{final}$ do
5:      for each $v_{source} \in \mathcal{V}$ do                ▷ Generate new traffic
6:          Generate $m \sim \mathcal{U}\left(1, max^{rob}\right)$ new robots with probability $p^{rate}$
7:          $\mathcal{M} \leftarrow \mathcal{M} \bigcup m$
8:      for each $agent \in T_i$ do                  ▷ Compute state-actions
9:          $agent.s \leftarrow$ count of robots on assigned edge $e$
10:         $agent.a \leftarrow \pi\ (agent.s)$                    ▷ $cost_e^{add}(t)$
11:     for each $e \in \mathcal{E}$ do                       ▷ Update graph costs
12:         $cost_e(t) \leftarrow cost_e^{travel} + cost_e^{add}(t)$          ▷ Equation (1)
13:     for each $m \in \mathcal{M}$ do
14:         if $m.edge\_transition$ then              ▷ Robot $m$ is waiting
                                             to transition to a new edge
15:             $m.path \leftarrow$ A* $(cost\ (t))$                ▷ Recompute path
16:         Increment robots along their path, obeying capacity constraints (Equation (6))
17:         $\mathcal{M} \leftarrow \mathcal{M} \setminus m.at\_goal$               ▷ Remove robots that
                                                       reach their goal
18:     Log $travel\ (t)$, $wait^{delay}\ (t)$, $wait^{enter}\ (t)$ according to robots in
        the system
19:     Compute $G\ (t)$                                     ▷ Equation (4)
20: return $G\left(t_{final}\right)$
```

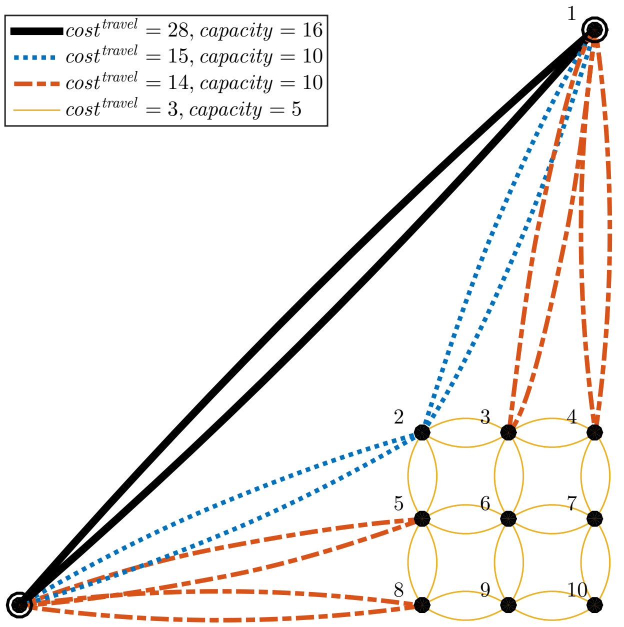

$cost^{travel} = 28$, $capacity = 16$
$cost^{travel} = 15$, $capacity = 10$
$cost^{travel} = 14$, $capacity = 10$
$cost^{travel} = 3$, $capacity = 5$
0
1
2
3
4
5
6
7
8
9
10

Fig. 3 The basic traffic graph that we consider. Robot traffic is randomly spawned at vertices 0 and 1, traveling to the opposite vertex, i.e. vertex 1 or vertex 0, respectively. Edges (0, 1), and (1, 0) are analogous to a "highway" between the two vertices. They have the highest capacity, carrying up to 16 robots each, and represent the shortest path, $cost^{travel} = 28$. The goal of the multiagent traffic management system is to incentivize traffic to take detours via the "small roads" to alleviate congestion

5 Experimental setup

In this section, we discuss the traffic simulation setup that we used to test our multiagent traffic management system. There are a number of parameters that we investigated in this work to probe the sensitivity of the system to different traffic profiles and graph structures. To control the parameters under scrutiny for each set of experiments, consider the basic traffic graph shown in Fig. 3.

For this traffic graph, the edges between vertices 0 and 1 are analogous to a "highway" between two sources of traffic. Over a single simulation epoch, vertex 0 and vertex 1 each randomly spawn robots that attempt to reach the opposite corner of the graph. That is, robots that spawn from vertex 0 aim to reach vertex 1 and vice versa. The shortest path between the two vertices is across the direct edges (0, 1) and (1, 0), however these edges have a fixed capacity of 16 and take 28 timesteps to cross. Thus, once an edge is at capacity, a newly spawned robot will have to wait for any of the existing traffic on that edge to reach their destination before it may enter. Alternatively, the robot may choose to take a different path across the graph, such as via edge (0, 2). This is analogous to taking a detour via the "small roads".

In this problem, it is clear that all robots choosing to take the shortest path will result in severe congestion along the "highway". The goal of the multiagent traffic management system is to apply additional costs to the graph that will incentivize traffic to avoid the "highway", and instead take the detours via vertices 2 through 10 to alleviate congestion. Notice, however, that the base travel costs of taking the "small roads" is higher than the direct "highway" route. In addition, the capacity of these edges is also lower than the "highway" edges. Thus, computing the magnitude of additional costs to apply that will successfully divert traffic is a non-trivial optimization problem, especially when coupled with a dynamic system with randomly spawning traffic.

Note that a robot that is ready to transition to a connecting edge may always replan its path given the latest graph cost data. For example, say a robot initially plans to travel along the sequence of vertices {0, 5, 2, 1}, however after traveling for 14 timesteps along edge (0, 5), it discovers that there is a now a large cost to traverse edge (2, 1). At this point, the robot can replan and divert its path along vertices {6, 3, 1}. Similarly, at any timestep that the robot is delayed from making an edge transition, it may continue to replan based on incoming data from the multiagent traffic management system. However, in our setup once a robot has entered an edge, it is committed to continuing along that edge until it is able to make its next transition. In each of the following experiments, and for all tested traffic management strategies, the robots use A* to find paths to their assigned goals that minimize traversal cost according to the current graph costs from Eq. (1).

Table 1 Simulation experiment sets for the source and sink traffic profile case study

Set	max^{rob}	p^{rate}
1	10	0.33
2	5	0.33
3	4	0.33
4	3	0.33
5	2	0.33
6	1	0.33
7	1	0.25
8	1	0.20
9	1	0.167
10	1	0.10

5.1 Case study: traffic profiles

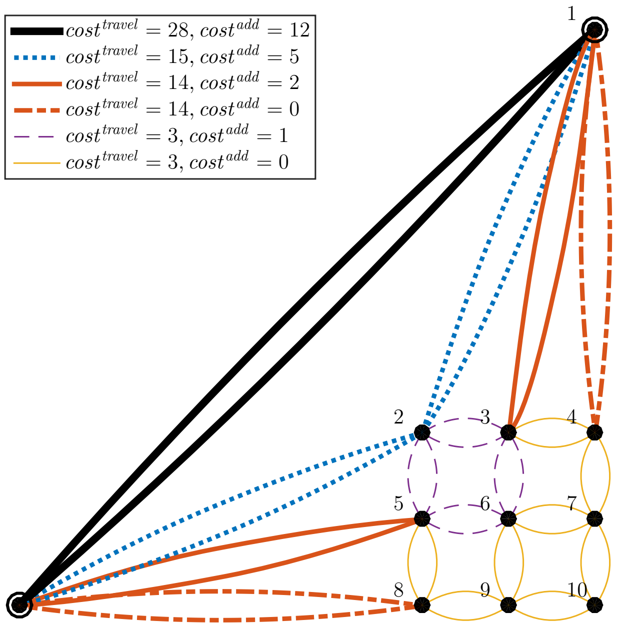
In our first case study, we explored the impact of different traffic profiles on the ability of our multiagent traffic management system to improve overall throughput. We tested two types of traffic profiles: a source and sink model, where all traffic is spawned at vertices {0, 1} going to vertices {1, 0}, respectively; and a distributed traffic profile, where traffic can be spawned from any vertex, going to any other vertex.

5.1.1 Source and sink traffic profile

For the source and sink traffic experiments, we investigated varying the severity and rate of traffic. To vary the traffic severity, we ran 6 sets of trials where at each timestep, up to $max^{rob} = \{10, 5, 4, 3, 2, 1\}$ new robots were spawned at each source vertex with a probability of $p^{rate} = 0.33$; that is, approximately every 3 timesteps, up to {20, 10, 8, 6, 4, 2} new robots were introduced into the system. To investigate the effect of varying traffic rates, we ran 4 additional sets of trials where, at each timestep, a new robot was spawned at either of the source vertices with a given probability $p^{rate} = \{0.33, 0.25, 0.20, 0.167, 0.10\}$. This can be thought of as spawning approximately 2 new robots in the system every {4, 5, 6, 10} timesteps. Table 1 summarizes the simulation experiments that we run for this case study.

In these experiments we compared the resulting traffic throughput against three baseline scenarios:

1. Robots only have access to the known traversal costs (*known costs*),
2. Robots have access to the expected traversal costs, i.e. fixed costs summed with expected delay due to other robots ahead in the queue to enter an edge (*expected costs*),
3. Robots observe fixed tolls in the system that are designed to equalize the cost of the four major routes between the two source and sink vertices (*fixed tolls*) (Arnott et al. 1993; Beria et al. 2015; Pigou 1920), see Fig. 4.



$cost^{travel} = 28, cost^{add} = 12$
$cost^{travel} = 15, cost^{add} = 5$
$cost^{travel} = 14, cost^{add} = 2$
$cost^{travel} = 14, cost^{add} = 0$
$cost^{travel} = 3, cost^{add} = 1$
$cost^{travel} = 3, cost^{add} = 0$
0
1
2
3
4
5
6
7
8
9
10

Fig. 4 The graph costs used for the comparative *fixed tolls* scenario. The $cost^{add}$ values for each edge were hand designed to approximately equalize the cost of traversing each of the four major routes between the source and sink vertices

We tested two versions of our proposed neuro-evolutionary multiagent traffic management system. One in which the edge traversal costs broadcast to the robots is the sum of the *known costs* and the output of the agent control policies (multiagent system with known costs). The second version sums the *expected costs* with the agent outputs (multiagent system with expected costs).

5.1.2 Distributed traffic profile

In addition to the source and sink traffic profile experiments, we also ran a series of experiments to investigate the performance of the multiagent traffic management system when exposed to a distributed traffic profile. In these experiments, robots were able to spawn from any vertex, going to any other vertex in the system. Given the more distributed nature of the traffic in these simulations, we only tested a subset of the higher traffic congestion scenarios at $max^{rob} = \{4, 5\}$ with $p^{rate} = 0.33$ (experiment sets 11 and 12, respectively).

Again, we compared against the *known costs* and *expected costs* scenarios described in the previous subsection. For this set of experiments, since robots were spawning randomly across the graph, there were no simple hand-designed tolls that could be generally applied to evenly distribute the traffic, and so we omit that set of comparative trials for this study.

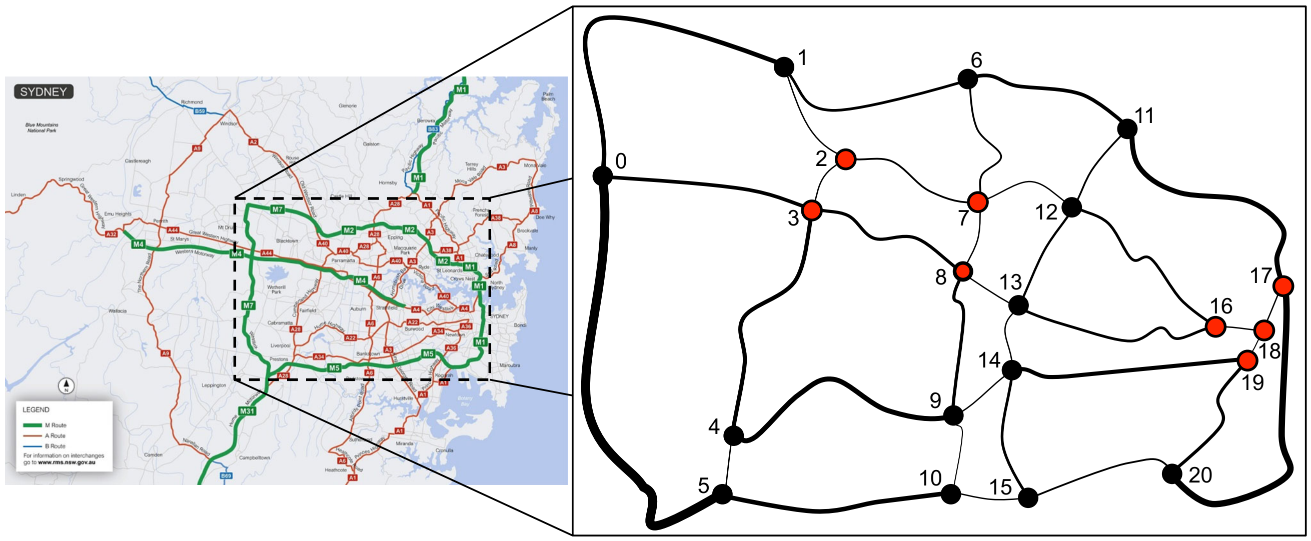


Fig. 5 The Sydney orbital network and adjoining highways [original underlying figure from NSW (2017b)]. The expanded inset shows the traffic graph we derived from the major motorways and highways in the network. The thickness of the edges represent the relative capacities of each edge. The red-centered vertices show the eight possible goal vertices. The cluster on the right is situated at four major intersections in Sydney city, and the cluster on the left consists of the four major highway exits into Parramatta (Color figure online)

5.2 Case study: Sydney orbital network

In our second case study, we investigated the performance of our proposed multiagent traffic management system on a real-world traffic domain. The robots in this system would be self-driving cars on the road network. For this set of experiments, we derived the traffic graph from the Sydney orbital network (NSW 2017c) and used data provided by the NSW Government Roads and Maritime Services to assign road capacities and the fixed travel costs for each edge in the graph (NSW 2017a). Specifically, we computed the capacity of each edge according to Level of Service (LOS) E which "describes operation at capacity. Operations at this level are volatile, because there are virtually no usable gaps in the traffic stream. Vehicles are closely spaced leaving little room to manoeuvre within the traffic stream. At capacity, the traffic stream has no ability to dissipate even the most minor disruption and any incident can be expected to produce a serious breakdown with extensive queuing (NSW 2017c)." We expect autonomously piloted vehicles to be able to operate at these capacities better than human-piloted vehicles and so we aim to design our traffic management system to also handle these conditions. The maximum density for LOS E is 28 passenger vehicles per kilometer per lane and for our experiments we round the edge capacities to the nearest 100 vehicles. See Fig. 5 and Appendix A for details on the full traffic graph configuration.

There are 21 vertices in the Sydney orbital network traffic graph and 70 directed edges. We tested a distributed traffic profile with 8 possible destination vertices situated around the two major central business districts in Sydney (Sydney city and Parramatta), see Fig. 5. This traffic profile is used to represent the typical morning rush hour traffic as commuters travel from various suburbs in the greater Sydney region to the main business districts for work. Given the larger number of source vertices, experiment sets 13 and 14 use traffic generation parameters $max^{rob} = 1$ and $p^{rate} = \{0.25, 0.2\}$, respectively.

6 Results

We first present the results from our case study on the basic traffic graph and analyze the observed learning performance and achieved traffic throughput trends with respect to varying traffic profiles. Following that, we present the results from our second case study on the Sydney orbital network traffic graph. Finally, we verify the performance of our trained multiagent traffic management system on a full physics simulation using the ROS-Gazebo environment. In the latter set of experiments, we apply the trained agents to manage the cost of travel for a set of five robots moving through a simulated warehouse environment. These results demonstrate that improvements to traffic throughput observed in the high level traffic graph experiments are transferable to a real-world traffic environment where robots may take a variable amount of time to cross each edge due to noisy actuation and previously unmodelled low level collision avoidance manoeuvres.

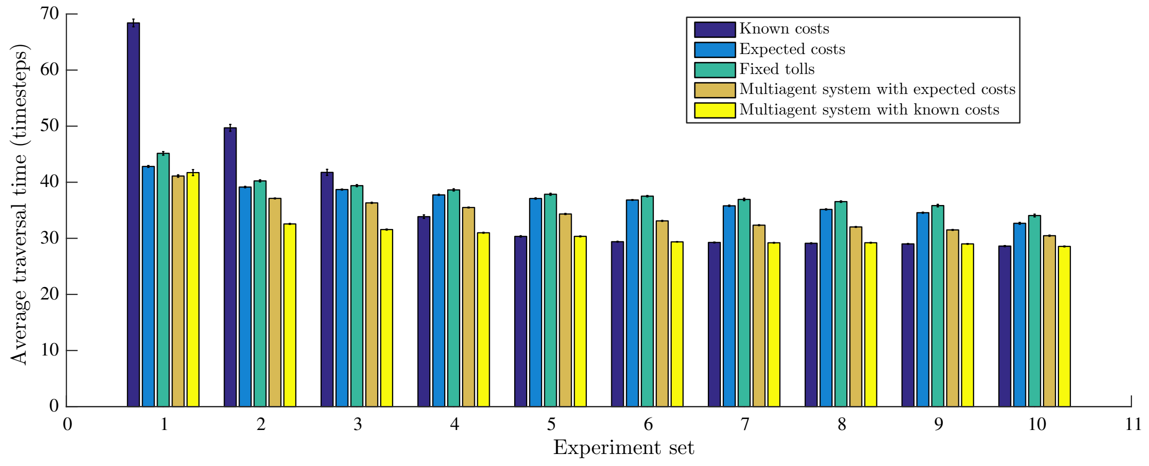


Fig. 6 Basic traffic graph: traffic throughput when robots plan on learned multiagent traffic management system costs, compared to planning on *known costs*, *expected costs*, and *fixed tolls*. The experiment sets correspond to those described in Table 1. Error bars show standard error in the mean over 20 statistical runs

6.1 Case study: traffic profiles

6.1.1 Source and sink traffic profile

The collated results from all ten experiment sets for the source and sink traffic profile, as outlined in Table 1, are shown in Fig. 6. The bar chart demonstrates the strong influence of traffic severity and rate on the average throughput with a general trend in decreasing travel time as the number of robots introduced into the system decreases. Both multiagent system variants (trained over 1000 learning epochs) produced superior performance when traffic severity was high (experiment sets 1 through 3), however as fewer robots were spawned and were spawned more infrequently, the multiagent system combined with the expected costs produced poorer throughput performance compared to having the robots plan purely on the known costs. The multiagent system combined with just the known costs produced the best performance overall, demonstrating that when the potential for congestion is high, the agents are able to influence the graph costs to effectively spread out the traffic. At the same time, when traffic is sparse, it allows the robots to take direct routes to their goal.

It is interesting to note that simply having the robots plan on the *expected costs* or *fixed tolls* can very effectively improve traffic throughput when congestion is high. However, the influence of these costs is simply to spread out the traffic. Thus, when traffic is sparse, this results in robots executing lengthier paths since spreading out is unnecessary. This effect is carried through to the multiagent system variant that combines the agent output costs with the expected costs, resulting in a small but persistent bias in the average travel time when compared to just using the *known costs* or the multiagent system outputs combined with the known costs.

In this set of experiments, each of the multiagent teams were trained over 1000 learning epochs. The performance of the multiagent traffic management system over sequential evolutionary generations is shown in Fig. 7. Both combinations of multiagent costs with known or expected costs are shown as the dashed or solid lines, respectively. In general, convergence is achieved after approximately 200 evolutionary epochs, with the system achieving faster convergence for lighter traffic profiles. Only in the highest traffic severity case (33% chance of up to 10 new robots at each spawn point at each timestep) did the CCEA require approximately 500 epochs to fully converge.

The results from the first six sets of experiments, investigating the effect of traffic severity, are shown in Fig. 7a, while the learning performance of the final five experiments, which vary the traffic rate, are shown in Fig. 7b. The plots show that there is a distinct benefit to only applying the known costs in addition to the agent output costs. Apart from the case with most severe traffic profile, the experiments in which the expected cost is incorporated produced travel times that were on average 15% longer than just using the known costs. The trade-off is that in cases of very severe traffic congestion ($max^{rob} = \{3, 4, 5, 10\}$ and $p^{rate} = 0.33$), using only the known costs results in very poor initial performance. These results show that the learned output costs of the multiagent system are able to compensate for the inadequacy of only using the known costs.

The plots in Fig. 7 also show that at traffic conditions lighter than $max^{rob} = 2$ and $p^{rate} = 0.33$, the multiagent system does not provide significant improvements to the overall traffic throughput. This is to be expected since at this level of traffic, the expected number of spawned robots at each source vertex is 14 every 28 timesteps (the time it takes to

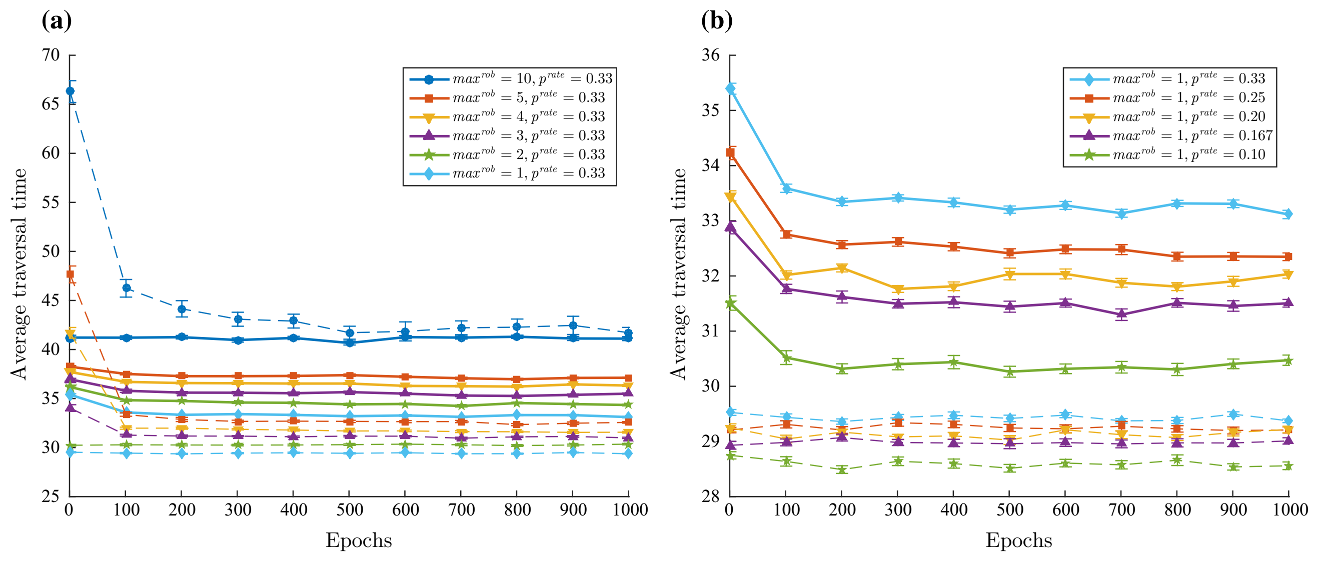


Fig. 7 Basic traffic graph: the progression in traffic management performance of the multiagent team over sequential evolutionary epochs. The solid lines plot the performance of the multiagent team combined with *expected costs* while the dashed lines plot the performance of the multiagent team combined with *known costs*, error bars show standard error in the mean over 20 statistical runs. The results for the first six experiment sets are shown in **a**, while the final five experiments are shown in **b**, note the differences in the *y* axis scale

cross the "highway"). Since the capacity of the highway edge is 16, then on average, all spawned robots are able to take the direct edge to their destination vertex and there is little need for the agents to incentivize diverse routes.

6.1.2 Distributed traffic profile

The results from the distributed traffic profile experiments are shown in Fig. 8. Since the multiagent traffic management system combined with known edge costs provided superior performance in the source and sink traffic profile experiments, we only tested this multiagent configuration with the distributed traffic profile. Given the learning profiles shown in Fig. 7, we also restricted the number of training epochs to 200 for this set of experiments. Furthermore, as noted in Sect. 5.1.2, since robots are spawning randomly across the graph, there exists no simple hand-designed tolls that can be generally applied to evenly distribute the traffic, and so we also omit the set of *fixed tolls* comparative trials for this study.

The general trends seen in Fig. 8 are similar to those observed in the source and sink traffic profile experiments. The multiagent traffic management system combined with known edge costs resulted in improved average traversal times when compared to robots planning purely on the *known costs* or the *expected costs*. Robots planning on the multiagent costs resulted in average traversal times that were 45.1% and 16.3% faster than planning only on the known costs (experiment sets 11 and 12, respectively). However, the difference between using the *expected costs* and those broadcast by the multiagent system was less pronounced in these experiments

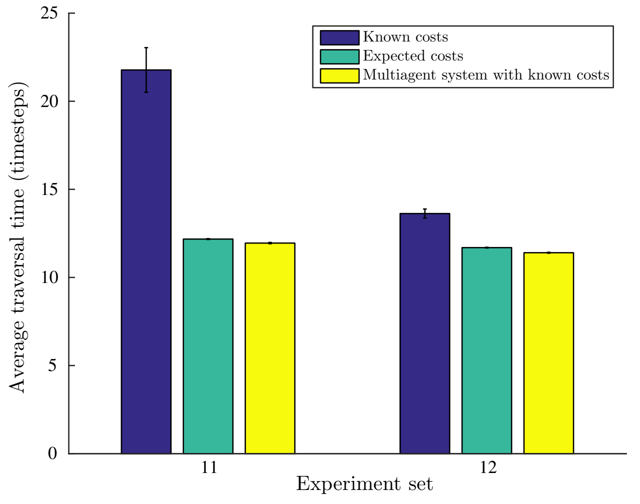


Fig. 8 Basic traffic graph: average traversal times for a distributed traffic profile where robots are spawned randomly across the graph and plan to randomly allocated goal vertices. Error bars show standard error in the mean over 20 statistical runs

(1.9% and 2.5% for experiment sets 11 and 12, respectively), suggesting that under a distributed traffic profile, applying any costing strategy that encourages diverse paths may be sufficient for alleviating congestion.

Additional traffic metrics are reported in Table 2. Columns 2–4 show the breakdown of total traffic traversal time into the percentage of time traffic spent moving along an edge (travel), waiting to transition from one edge to another [wait (delay)], and waiting to enter the system from an origin ver-

Table 2 Traffic metrics from the distributed traffic simulation experiments

	%travel	%wait (delay)	%wait (enter)	# robots	%goal reached
Experiment set 11					
Known costs	44.7 ± 4.4	16.6 ± 8.4	38.7 ± 25.6	2164.3 ± 93.8	92.7 ± 0.7
Expected costs	94.2 ± 3.4	3.4 ± 0.4	2.4 ± 0.4	2134.8 ± 73.3	94.3 ± 0.6
Multiagent system	94.7 ± 3.2	2.8 ± 0.5	2.4 ± 0.5	2116.3 ± 53.1	94.3 ± 1.0
Experiment set 12					
Known costs	83.7 ± 2.7	7.4 ± 3.1	8.9 ± 6.6	1796.5 ± 67.1	93.5 ± 0.7
Expected costs	97.5 ± 2.9	1.5 ± 0.3	1.0 ± 0.2	1813.2 ± 58.9	94.4 ± 0.6
Multiagent system	97.3 ± 2.0	1.5 ± 0.3	1.2 ± 0.3	1766.0 ± 33.8	94.8 ± 0.8

tex [wait (enter)]. These results also support the hypothesis that encouraging diverse paths can significantly reduce congestion. Robots that planned on either the expected costs or the multiagent costs spent far less time waiting and substantially more time moving towards their goal compared to those robots that planned on the known costs.

6.2 Case study: Sydney orbital network

For this case study, we again assessed the traffic throughput when robots planned using the known costs, the expected costs or the combined costs from the multiagent system with known costs. The results after 200 learning epochs are shown in Fig. 9. As demonstrated in the previous case study, the multiagent traffic management system out-performed all other fixed-cost methods. Robots that planned on the multiagent costs resulted in 32.5% and 27.1% faster traversal times compared to those that planned purely on the known costs, they were also 6.6% and 4.8% faster when compared to those that planned on the expected costs (experiment sets 13 and 14, respectively).

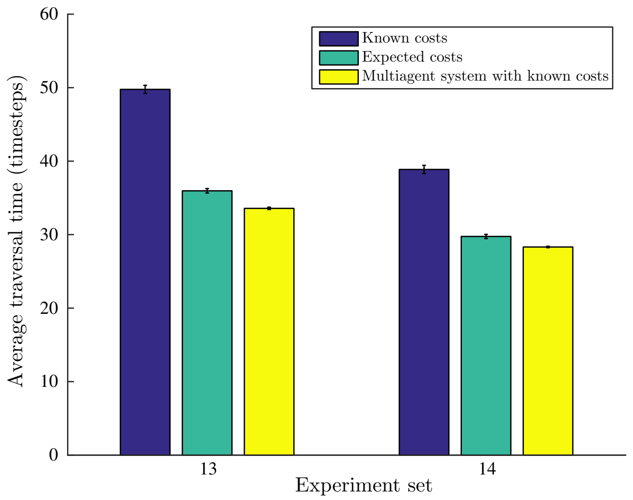


Fig. 9 Sydney orbital network: average traversal times for a distributed traffic profile where robots are spawned randomly across the graph and plan to one of eight randomly allocated goal vertices. Error bars show standard error in the mean over 20 statistical runs

The additional traffic metrics, given in Table 3, show that robots that planned on the multiagent system costs spent less time waiting, either to enter the system or while delayed en route. In total, robots that used the multiagent system costs spent 38.9% of their time waiting compared to 67.9% (known costs) and 45.6% (expected costs) for experiment set 13. For the lighter traffic profile in experiment set 14, these numbers are 23.1% (multiagent system) compared to 49.3% (known costs) and 28.3% (expected costs). The difference in the distribution of total traversal time between Tables 2 and 3 shows that in a more complex domain such as the Sydney orbital network, applying diverse routes can still result in significant delay and congestion in the system.

6.3 ROS-Gazebo experiments

The simulated warehouse environment constructed in Gazebo and used in the following experiments is shown in Fig. 10. The graph topology of this environment is equivalent to the basic traffic graph shown in Fig. 3, however, given the relative size of the robots and the available free space, we reduced the capacity of the edges such that all edges stemming from vertices 0 or 1 had a capacity of 2 robots, except for those edges that connected to vertex 2 for which the capacity was set to 1 robot. All other edges also had a capacity of 1 robot.

As in the earlier case studies. the multiagent traffic management team was first trained via CCEA in the high level simulator using only the high level graph abstraction of the problem. The evolved neural network weights that defined the champion team at the end of learning were then output and stored for later use in the ROS-Gazebo simulator.

We combined several of the basic ROS packages for localization (Open Source Robotics Foundation 2017) and waypoint navigation (Open Source Robotics Foundation 2016) with our custom packages for instantiating the distributed multiagent traffic management team as well as the robot planning routines over the high level traffic graph. All

Table 3 Traffic metrics from the Sydney orbital network simulation experiments

	%travel	%wait (delay)	%wait (enter)	# robots	%goal reached
Experiment set 13					
Known costs	32.1 ± 0.8	31.2 ± 3.0	36.7 ± 5.5	1039.3 ± 36.2	83.2 ± 0.5
Expected costs	54.4 ± 1.5	22.0 ± 1.8	23.6 ± 4.0	$1045.5.3 \pm 33.6$	86.1 ± 0.7
Multiagent system	61.1 ± 1.3	21.3 ± 1.6	17.6 ± 2.6	1022.2 ± 21.0	85.5 ± 0.9
Experiment set 14					
Known costs	50.7 ± 1.4	29.1 ± 3.9	20.2 ± 4.8	826.5 ± 25.5	82.9 ± 0.9
Expected costs	71.7 ± 1.6	18.7 ± 2.5	9.6 ± 3.0	824.9 ± 20.9	86.1 ± 0.8
Multiagent system	76.9 ± 1.7	16.9 ± 2.0	6.2 ± 1.5	809.6 ± 19.6	86.2 ± 0.7

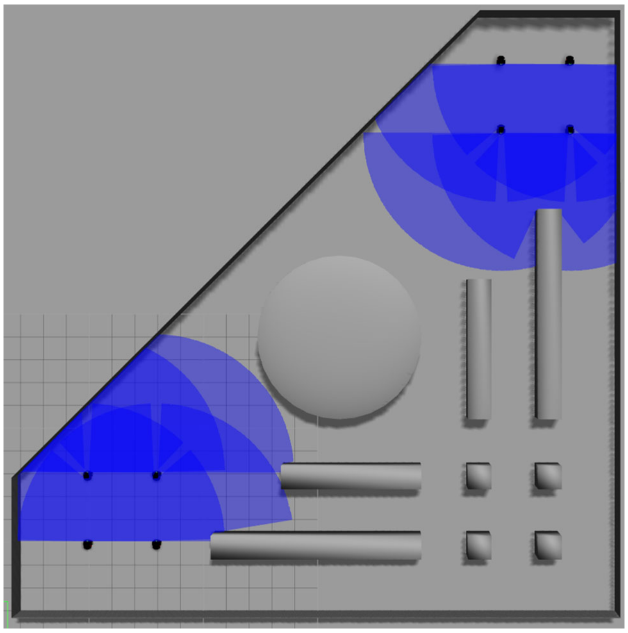


Fig. 10 Aerial view of the Gazebo warehouse environment in which we tested our multiagent traffic management system. Each Pioneer P3-DX robot is equipped with a 180° field of view 2D lidar, range-limited to 6 m. Robots are tasked to deliver packages between the top right and bottom left of the simulation world

packages necessary to run these experiments can be found at Chung (2018). In these experiments, we tested the scenario with eight simulated Pioneer P3-DX robots initially in the configuration shown in Fig. 10. Each robot was tasked to travel between the two diagonal corners of the world, for example to deliver packages, simulating the source and sink scenario described in Sect. 5.1.1. We report the number of delivery missions completed by the robots and the average time for each delivery in Table 4. We tested robot team sizes of five and eight robots over a fixed time period of 5 and 11 min, respectively, to see how well the multiagent system scales to high levels of congestion. In these experiments, the robot velocities were scaled to a maximum velocity of 0.35 m/s. Thus, executing the straight line path for each robot (ignoring obstacles) would take at least 68.7 s. Using the multiagent traffic management system, we are able to achieve an average traversal time of 72.6 s for the five robot team, and 78.5 s for the eight robot team.

Table 4 ROS-Gazebo experiments with eight robots delivering packages between two fixed locations in the warehouse

	# robots	With agents	No agents
Average time	5	72.6 s	81.6 s
Deliveries in 5 min	5	27	16
Average time	8	78.5 s	95.5 s
Deliveries in 11 min	8	61	51

As a comparison, we simulated the same set of delivery missions in the absence of traffic management. In this latter experiment, the robots completed significantly fewer delivery missions; 16 compared to 27 for the five robot team, and 51 compared to 61 for the eight robot team. In addition, there were two cases during the latter experiment when a robot was forced to abandon its delivery mission and turn back to its previous waypoint due to congestion levels that their low level collision avoidance planners were unable to handle. See Fig. 11 for a comparison between the paths executed by the robots with and without the multiagent traffic managers. For those missions that were successful, the average traversal time was significantly higher at 81.6 s and 95.5 s for the five and eight robot experiments, respectively.

7 Conclusions and future work

Massively distributed autonomous traffic networks provide a challenging optimization problem in terms of routing robots to maximize throughput while maintaining safety requirements. In this work, we applied a multiagent neuro-evolutionary approach to train a team of traffic management agents to manipulate the cost space in which the robots planned their paths. Our results demonstrate that the multia-

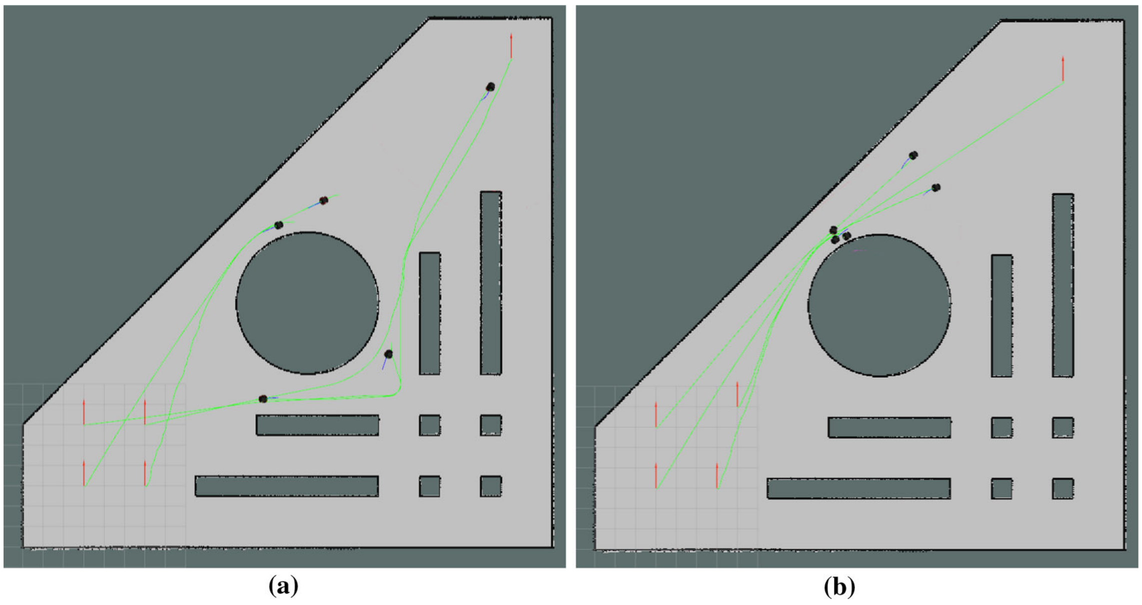


Fig. 11 Snapshot of the robots and their planned paths at approximately 100 s into the experiment. In **a** the multiagent costs incentivize the robots to plan paths that go around the bottom and right side of the circular obstacle even though they may be longer than going around the left of the obstacle. In comparison, **b** shows the resulting paths when no traffic management is applied and robots greedily plan and attempt to execute their shortest paths. The robots tend to get stuck while passing through the narrow corridor

gent team incentivizes robots to take diverse paths. Although these paths may be considered suboptimal in terms of distance, robots that executed these paths not only completed more successful traversals, but they also resulted in faster average traversal times when compared to robots that planned solely on the distance costs. This is because the multiagent team was able to synthesize data from a range of traffic profiles and learn to adjust the costs to account for potential congestion that would cascade throughout the system.

The current work uses a direct implementation of CCEA where each agent in the team trains directly based on the global team reward. This assumes that all agents in the team contributed equally, or indeed, have the ability to contribute equally to the final reward. However, we can see in some cases, such as in the basic traffic graph, that this is clearly not the case. The agents managing traffic along the diagonal edges have a higher impact on the system performance, especially in the case of the source and sink traffic profile. For a distributed traffic profile, it is not as simple to analyze since we cannot easily point to an edge, or set of edges, that represent the system bottleneck. An interesting direction of future research is to incorporate this *impactfulness* analysis into the training such that each agent can receive a more accurate and precise reward signal. Combining this more generally with ideas regarding the structural credit assignment problem in multiagent learning can provide unique solutions to the challenges of concurrent learning in large distributed multiagent systems.

The neural network control policies that we use in this work have a very basic architecture. This design choice was motivated by our aim to demonstrate the ability of the multiagent team to learn complex traffic management strategies despite individual agents only having access to local information. We believe that more sophisticated network structures with access to more traffic information, such as neighboring congestion states, could potentially capture more complex patterns in the traffic flow. However, we are also cognizant of the learning challenges that arise from increasing the dimensionality of the state-action space. Nevertheless, we believe that this would be a valuable and promising area for future work.

Finally, in our experiments, we have only explored a limited set of robot planning behaviors. Namely, the robots use A* to plan paths according to the current known costs, where those costs are defined by the agents. Furthermore, robots were triggered to replan en route whenever they finished traversing their current edge. These behaviors are not made explicit to the multiagent traffic management team, indeed the multiagent framework is agnostic to the underlying planners used by the robots; however, this does limit the scope of traffic profiles that are observed during training, which in turn can hinder the performance of the trained agents when managing robots that use different path planning protocols.

The fundamental assumption that we make is that the agents are able to exert some influence over the robot plans through their actions, that is, we assume that the robots' planning objectives are a function of the graph costs assigned by the agents. The controllability of the traffic system can be described as the sensitivity of the robot motion to the actions of the agents. For example, compare the current case where robots replan at edge transitions to one where robots must execute their first plan. The controllability of these two systems is quite different. In general, we expect the ability of the multiagent team to learn an effective traffic management strategy to decrease as controllability decreases. The extreme case of this is where all robots in the system are non-compliant, that is, where they ignore the traversal costs assigned by the agents. It would be a valuable exercise to investigate the change in traffic management performance as controllability is reduced as this could lead to strategies that are more robust to non-compliance as well as stochasticity and uncertainty in the traffic environment.

Appendix A Sydney orbital network traffic graph configuration

The fixed travel cost of each edge is the distance of each highway section between the defined vertices rounded to the nearest kilometer. Edges with common end vertices are taken to have the same travel costs and capacities, e.g. edges $(0, 1)$ and $(1, 0)$ have the same parameters and so only edge $(0, 1)$ is listed below (Table 5).

Table 5 Sydney orbital network traffic graph configuration

v_0	v_1	$cost^{travel}$	$capacity$
0	1	18	10
0	3	10	8
0	5	35	20
1	2	5	3
1	6	10	6
2	3	3	3
2	7	8	4
3	4	13	9
3	8	8	7
4	5	3	3
4	9	13	11
5	10	12	10
6	7	7	4
6	11	9	8
7	8	3	3
7	12	5	3
8	9	8	7
8	13	3	3
9	10	4	2
9	14	4	3
10	15	4	3
11	12	5	4
11	17	13	11
12	13	6	5
12	16	10	6
13	14	4	3
13	16	11	6
14	15	7	6
14	19	13	9
15	20	7	4
16	18	3	3
17	18	3	3
17	20	17	14
18	19	2	1
19	20	8	7

References

Agogino, A. K., & Tumer, K. (2004). Efficient evaluation functions for multi-rover systems. In *Genetic and evolutionary computation conference* (pp. 1–11).

Agogino, A. K., & Tumer, K. (2012). A multiagent approach to managing air traffic flow. *Autonomous Agents and Multi-Agent Systems*, *24*(1), 1–25.

Arnott, R., De Palma, A., & Lindsey, R. (1993). A structural model of peak-period congestion: A traffic bottleneck with elastic demand. *The American Economic Review*, *83*, 161–179.

Arnott, R., & Kraus, M. (1998). When are anonymous congestion charges consistent with marginal cost pricing? *Journal of Public Economics*, *67*(1), 45–64.

Aubert, M. C., Üzümcü, S. C., Hutchins, A. R., & Cummings, M. L. (2015). Toward the development of a low-altitude air traffic control paradigm for networks of small, autonomous unmanned aerial vehicles. In *AIAA infotech @ aerospace* (pp. 1110–1117).

Baskar, L. D., De Schutter, B., Hellendoorn, J., & Papp, Z. (2011). Traffic control and intelligent vehicle highway systems: A survey. *IET Intelligent Transport Systems*, *5*(1), 38–52.

Bellemans, T., De Schutter, B., & De Moor, B. (2006). Model predictive control for ramp metering of motorway traffic: A case study. *Control Engineering Practice*, *14*(7), 757–767.

Beria, P., Ramella, F., & Laurino, A. (2015). Motorways economic regulation: A worldwide survey. *Transport Policy*, *41*, 23–32.

Bongiorno, C., Gurtner, G., Lillo, F., VAlori, L., Ducci, M., Monechi, B., & Pozzi, S. (2013). An agent based model of air traffic management. In *Proceedings of the 3rd SESAR innovation days*. Stockholm.

Bullo, F., Frazzoli, E., Pavone, M., Savla, K., & Smith, S. L. (2011). Dynamic vehicle routing for robotic systems. *Proceedings of the IEEE*, *99*(9), 1482–1504.

Carey, M., & Srinivasan, A. (1993). Externalities, average and marginal costs, and tolls on congested networks with time-varying flows. *Operations Research*, *41*(1), 217–231.

Chung, J. J. (2018). Git repository. https://github.com/JenJenChung. Accessed 21 February 2018.

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.

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.

Emami, H., & Derakhshan, F. (2012). An overview on conflict detection and resolution methods in air traffic management using multi agent systems. In *16th CSI international symposium on artificial intelligence and signal processing (AISP)* (pp. 293–298). IEEE.

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.

Gawrilow, E., Köhler, E., Möhring, R. H., & Stenzel, B. (2008). Dynamic routing of automated guided vehicles in real-time. In H.-J. Krebs & W. Jäger (Eds.), *Mathematics—Key technology for the future* (pp. 165–177). Berlin: Springer. (**Chapter 5**).

Hegyi, A., De Schutter, B., & Hellendoorn, H. (2005a). Model predictive control for optimal coordination of ramp metering and variable speed limits. *Transportation Research Part C: Emerging Technologies*, *13*(3), 185–209.

Hegyi, A., De Schutter, B., & Hellendoorn, J. (2005b). Optimal coordination of variable speed limits to suppress shock waves. *IEEE Transactions on Intelligent Transportation Systems*, *6*(1), 102–112.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.

Huang, P.-C., Lehman, J., Mok, A. K., Miikkulainen, R., & Sentis, L. (2014). Grasping novel objects with a dexterous robotic hand through neuroevolution. In *IEEE symposium on computational intelligence in control and automation (CICA)* (pp. 1–8). IEEE.

Kopardekar, P. (2015). *Safely enabling low-altitude airspace operations: Unmanned aerial system traffic management (UTM)*. Technical Report ARC-E-DAA-TN22234, NASA, April 2015.

Kopardekar, P., Rios, J., Prevot, T., Johnson, M., Jung, J., & Robinson, J. E. III (2016). Unmanned aircraft system traffic management (UTM) concept of operations. In *AIAA aviation technology, integration, and operations conference* (pp. 1–16).

Lindsey, C. R., & Verhoef, E. T. (2000). Traffic congestion and congestion pricing. In *Handbook of transport systems and traffic control* (pp. 77–105).

Land Transport Authority of Singapore (2017). Electronic road pricing (ERP). https://www.lta.gov.sg/content/ltaweb/en/roads-and-motoring/managing-traffic-and-congestion/electronic-road-pricing-erp.html. Accessed 11 October 2017.

NSW Government Roads and Maritime Services (2017a). Motorway design guide: Capacity and flow analysis. http://www.rms.nsw.gov.au/business-industry/partners-suppliers/documents/motorway-design/motorway-design-guide-capacity-flow-analysis.pdf. Accessed: 22 December 2017.

NSW Government Roads and Maritime Services (2017b). Sydney route number map. http://www.rms.nsw.gov.au/documents/roads/using-roads/alpha-numeric/sydney-map.jpg. Accessed: 22 December 2017.

NSW Government Roads and Maritime Services (2017c). Your guide to using the Sydney motorway network. http://www.rms.nsw.gov.au/sydney-motorways/documents/sydney-motorways-map.pdf. Accessed: 22 December 2017.

Open Source Robotics Foundation. (2016). move_base. http://wiki.ros.org/move_base. Accessed 21 February 2018.

Open Source Robotics Foundation (2017). amcl. http://wiki.ros.org/amcl. Accessed 21 February 2018.

Panait, L., & Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, *11*(3), 387–434.

Papageorgiou, M., Hadj-Salem, H., & Middelham, F. (1997). ALINEA local ramp metering: Summary of field results. *Transportation Research Record: Journal of the Transportation Research Board*, *1603*, 90–98.

Pechoucek, M., & Sislak, D. (2009). Agent-based approach to free-flight planning, control, and simulation. *IEEE Intelligent Systems*, *24*(1), 14–17.

Pigou, A. C. (1920). *The economics of welfare*. London: Macmillian and Co.

Prevot, T., Homola, J. R., Martin, L. H., Mercer, J. S., & Cabrall, C. D. (2012). Toward automated air traffic controlinvestigating a fundamental paradigm shift in human/systems interaction. *International Journal of Human–Computer Interaction*, *28*(2), 77–98.

Qiu, L., Hsu, W.-J., Huang, S.-Y., & Wang, H. (2002). Scheduling and routing algorithms for AGVs: A survey. *International Journal of Production Research*, *40*(3), 745–760.

Rebhuhn, C., Skeele, R., Chung, J. J., Hollinger, G. A., & Tumer, K. (2015). Learning to trick cost-based planners into cooperative behavior. In *2015 IEEE/RSJ international conference on intelligent robots and systems* (pp. 4627–4633).

Rossi, F., Zhang, R., Hindy, Y., & Pavone, M. (2018). Routing autonomous vehicles in congested transportation networks: Structural properties and coordination algorithms. *Autonomous Robots*, *42*, 1–16.

Stephanedes, Y. J., Kwon, E., & Chang, K. (1992). Control emulation method for evaluating and improving traffic-response ramp metering strategies. *Transportation Research Record*, *1360*, 42–45.

Taghaboni-Dutta, F., & Tanchoco, J. M. A. (1995). Comparison of dynamic routeing techniques for automated guided vehicle system. *International Journal of Production Research*, *33*(10), 2653–2669.

Tomlin, C., Pappas, G. J., & Sastry, S. (1998). Conflict resolution for air traffic management: A study in multiagent hybrid systems. *IEEE Transactions on Automatic Control*, *43*(4), 509–521.

Yliniemi, L., Agogino, A. K., & Tumer, K. (2014). Evolutionary agent-based simulation of the introduction of new technologies in air traffic management. In *Proceedings of the 2014 annual conference on genetic and evolutionary computation* (pp. 1215–1222).

Jen Jen Chung is a Senior Researcher in the Autonomous Systems Lab at ETH Zürich. Her current research interests include multi-robot coordination methods and risk-aware planning for aerial and ground robots. Her past research includes multiagent learning methods at Oregon State University and information-based exploration–exploitation strategies for autonomous soaring platforms at the Australian Centre for Field Robotics in the University of Sydney. She received her Ph.D. (2014) and B.E. (2010) from the University of Sydney.

Carrie Rebhuhn is a Senior Autonomous Intelligent Systems Engineer at MITRE. She completed her Ph.D. with a minor in Computer Science (2017) at Oregon State University. Her research focused on ways to coordinate robot trajectories in an obstacle-filled environment. She received her M.S. (2013) and B.S. (2011) in Mechanical Engineering from Oregon State University.

Connor Yates is a Robotics Ph.D. student at Oregon State University in the Collaborative Robotics and Intelligent Systems (CoRIS) Institute. His primary research focuses on multiagent coordination in complex tasks, and representing and explaining the intent of autonomous robotic teams. He received his B.S. in Computer Science from Oregon State University where he focused on leveraging intent in multiagent decision making.

Geoffrey A. Hollinger is an Assistant Professor in the School of Mechanical, Industrial and Manufacturing Engineering at Oregon State University. His current research interests are in planning, decision making, and learning for robotic systems in marine, air, and ground environments. He has served as PI and co-PI on research grants from ONR, NSF, USDA, and NASA. His past research includes networked marine robotics at the University of Southern California, multi-robot search at Carnegie Mellon University, personal robotics at Intel Research Pittsburgh, active estimation at the University of Pennsylvania's GRASP Laboratory, and miniature inspection robots for the Space Shuttle at NASA's Marshall Space Flight Center. He received his Ph.D. (2010) and M.S. (2007) in Robotics from Carnegie Mellon University and his B.S. in General Engineering along with his B.A. in Philosophy from Swarthmore College (2005). He has been a Guest Editor for the Autonomous Robots journal, Associate Editor for the ICRA and IROS conferences, and Area Chair for the RSS conference.

Kagan Tumer is a Professor of Robotics, and the Director of the Collaborative Robotics and Intelligent Systems Institute at Oregon State University. Dr. Tumer's research interests are control, coordination and optimization in large complex systems with a particular emphasis on multiagent coordination. Applications of his work include coordinating multiple robots, optimizing large sensor networks, controlling autonomous vehicles, reducing traffic congestion and managing air traffic. His work has led to over one hundred and fifty publications, including three edited books, one patent, and several best paper awards. He is an associate editor of the Journal on Autonomous Agents and Multiagent Systems, and was the program co-chair of the 2011 Autonomous Agents and Multiagent Systems Conference (AAMAS 2011). Dr. Tumer received his Ph.D. (1996) in Electrical and Computer Engineering at The University of Texas, Austin, and is a member of AAAI and a senior member of IEEE.