Exploring the potential of multiagent learning for autonomous intersection management

Matteo Vasirani
Artificial Intelligence Group
University Rey Juan Carlos
Madrid, Spain
matteo.vasirani@urjc.es

Sascha Ossowski
Artificial Intelligence Group
University Rey Juan Carlos
Madrid, Spain
sascha.ossowski@urjc.es

ABSTRACT

The problem of advanced intersection management is being discovered as a promising application field for multiagent technology. In this context, drivers interact autonomously with a coordination facility that controls the traffic flow through an intersection, with the aim of avoiding collisions and minimizing delays. This is particularly interesting for the case of autonomous vehicles that are controlled entirely by agents, a scenario that will become possible in the near future.

In this paper, we seize the opportunities for multiagent learning offered by such a scenario, by introducing a coordination mechanism where teams of agents decentrally coordinate their velocities when approaching the intersection. We show that this approach enables the agents to improve the intersection efficiency, by reducing the average travel time and so alleviating traffic congestions.

1. INTRODUCTION

Traffic congestion is a costly problem of cities in all developed countries. Many human-centered instruments and solutions (e.g. message signs, temporary lane closings, maximum velocity changes), are deployed in highways and roads in order to speed up the traffic flow. Nevertheless, in line with the recent advances of telematic infrastructures, the problem of road traffic management is being discovered as a promising application field for multiagent technology [1]. Multiagent systems (MAS) are the ideal candidates for the implementation of road traffic management systems, due to the intrinsically distributed nature of traffic-related problems.

In this context, the problem of advanced intersection management, where drivers interact autonomously with a coordination facility that controls the traffic flow through an intersection so as to avoid collisions while minimizing delays, is receiving more and more attention.

In [2] is presented a reservation-based system in which vehicles request an intersection manager to reserve the necessary time slots during which they may pass through the intersection. This work opens many possibilities for multiagent learning, with the goal of improving the efficiency of intersections.

In this paper, we present a coordination mechanism based on Probability Collectives (PC) [10]. With such an approach, teams of agents decentrally coordinate their velocities during their approximation to the intersection, with the aim of reducing the average travel time by making better, non-conflicting, reservations.

The paper is structured as follows: in section 2 we present the intersection management problem, as proposed by Dresner and Stone [2, 3]; in section 3 we introduce our proposal for making the agents learn to coordinate their actions in order to improve the intersection efficiency; in section 4 we show the results of the experiments and we discuss the possibility of improving them. Finally in section 5 conclusions and future work is outlined.

2. RESERVATION-BASED INTERSECTION MANAGEMENT

The reservation-based system proposed in [2] consists of two different kind of agents: intersection managers and driver agents. An intersection manager is responsible for managing the vehicles that want to pass through the intersection, by assigning the necessary time slots; a driver agent is responsible for controlling the vehicles to which it is assigned.

Each driver agent, when approaching the intersection, "calls ahead" the intersection manager and requests a reservation of space and time in the intersection. Such a request contains the necessary information to simulate the vehicle journey through the intersection, such as the vehicle properties (vehicle ID, vehicle size,...) as well as some properties of the proposed reservation (arrival time, arrival velocity, type of turn, arrival lane, arrival road segment,...).

The intersection manager then determines whether or not a request is feasible, by checking the confirmed reservations that are stored in its database. If the request is confirmed by the intersection manager, the driver agent stores the reservation details and tries to meet them. Otherwise, it slows down and makes another request at a later time.

The reservation system offers many opportunities for improving the efficiency of intersection, by incorporating learning mechanisms in the agents of such scenario [4]. For example, since the intersection manager serves the requests in a "first-come-first-served" fashion, it is possible to relax this constraint and allow the intersection manager to respond to requests at a later time. In this way the intersection manager can evaluate more competing requests at the same time and make a more well-informed decision.

While the learning opportunities for the intersection manager are of the form of *single agent learning*, the very *multi-*

agent learning opportunities reside in the driver agents. In the current implementation, driver agents must estimate the arrival time at the intersection, the arrival velocity, the arrival lane ... without communication nor coordination with the other driver agents; each agent makes its request on the basis of its actual velocity, and, if the request is rejected, the driver slows down and tries again. On the other hand, by letting the agents form teams and coordinate their actions, we provide them with more information that they use to make decisions.

2.1 Intersection model

We started from the model of intersection management proposed by Dresner and Stone [2, 3]. A driver agent cannot cross the intersection without a confirmed reservation. For this reason, we assume that when it reaches a minimum distance to the intersection, it is obliged to start making reservation requests. If a driver agents arrives at the intersection without a confirmed reservation, it stops and keeps trying to find a time slot when it may pass. The fullfilment or not of this norm is not guaranteed by any authority. Simply it is assumed that the driver agents are rational and that it is not convenient for them to risk a crash by crossing the intersection without a confirmed reservation.

If a request is confirmed by the intersection manager, the driver agent continuosly checks if it can meet the request conditions. If it realizes that is not able to meet them, due to the traffic conditions, it cancels its reservation by sending a message to the intersection manager, and prepares a new reservation request. Again, doing so is in its interest because a driver agent can only have one reservation, and a confirmed reservation that is not possible to meet is useless to hold.

2.2 Agent model

Our environment imposes a series of constraints on the agent behaviour. We assume that each driver agent has a preferred velocity which tries to keep along the entire journey. We also assume that when a vehicle appears in a lane of a road segment, it cannot change it during the approach to the intersection[1]. In this way, if a front vehicle proceeds at a lower velocity, the following vehicle is obliged to slow down. Furthermore, as demonstrated in [3], it is not convenient that the driver agents could turn from any lane, so in our model turning right (respectively left) is only possible from the rightmost (respectively leftmost) lane of a road segment.

The actions that a driver agent can autonomously take are related to the velocity at which it crosses the intersection. In particular, an agent could set its velocity to a value in the (discretized) interval $[1, preferredVelocity]$.

So, for the generic driver agent a_i, the variable x_i that defines its action is

$$x_i = \langle vehicleID, direction, lane, \\ turn, arrivalTimeAtIntersection, \\ arrivalVelocityAtIntersection \rangle$$

The field $arrivalTimeAtIntersection$ is implicitly set by the specific $arrivalVelocityAtIntersection$, while the fields $vehicleID$, $direction$, $lane$ and $turn$ are constant parameters.

[1]This is a feature that we plan to remove from the model in the future.

We assume that there are no misunderstandings regarding the ontology that describes the geometric configuration of the intersection, e.g. the lane 3 along the *North* direction corresponds to the same physical lane for every vehicle.

3. LEARNING TO COORDINATE

3.1 Global objective

To improve the efficiency of the intersection, we take the perspective of a system designer, whose goal is minimizing the travel time of the vehicles. The travel time for the generic driver agent a_i depends not only on its velocity while crossing the intersection, but also on the conflicts that may occur among different competing requests.

Let $\mathcal{C}$ be a set of driver agents, $\mathcal{C} = \{a_1, a_2, \ldots, a_n\}$. Each agent can take an action of the form defined in section 2.2. So, the vector $x = \langle x_1, x_2, \ldots, x_n \rangle$ defines the joint action of this set of agents. A possible function [2] that rates "how good" is a joint action, from the system designer perspective, is

$$G(x) = (1 + P(x)) \cdot D(x) \quad (1)$$

where $P(x)$ is the number of collisions resulting from the full joint action x, and $D(x)$ is the time spent by the agents to cross the intersection. We remark that a generic joint action x contains all the necessary information to simulate the agent journeys through the intersection, in the same way it is done by the intersection manager, so that is is also possible to calculate the number of conflicts among them as well as the travel time.

3.2 Agent private utility

The multiagent learning challenge here is making agents learn to act in an environment that is not merely a black-box that produces a reward for every action taken by the agent, but it is actually composed of other learning agents, i.e. the reward an agent receives for its actions depends also on the actions of other agents. So there is a strict relation between the private utility function of a single agent and the global objective of the system.

A recent advance in this direction is that proposed by the COllective INtelligence (COIN) [7, 8, 9] framework. The aim of COIN is studying the properties that a utility function of a learning agent situated in a multiagent environment must meet. COIN introduced the concepts of *factoredness* and *learnability* of an agent private utility function. A private utility function g_i is meant to be *factored* if it is aligned with the global utility G, i.e. if the private utility increases, the global utility does the same. Furthermore, it has to be easily *learnable*, i.e. it must enable the agent to distinguish its contribution to the global utility from that of the other agents. For example, the *Team Games Utility*, $TGU_i(x) = G(x)$, is trivially *aligned*, but is poorly *learnable*. If for example agent a_i takes an action that actually improves the global utility, while all the other agents take actions that worsen the global utility, agent a_i wrongly believes that its action was bad.

[2]It is possible to formulate other objective functions that take in consideration different relationship between collisions and time, as well as including other aspects, such as congestion or lane changes.

Better results have been obtained [9] with the *Difference Utility* (DU), defined as follows:

$$DU_i(x) = G(x) - G(CL_i(x)) \quad (2)$$

where x if the joint action of the collective, $G(x)$ is the global utility derived from such joint action, and $G(CL_i(x))$ is the "virtual" joint action formed by replacing with a constant factor c all the components of x affected by agent a_i. If this constant is $\overrightarrow{0}$, i.e. the null action, the DU is equivalent to the global utility minus the global utility that would have arisen if the agent a_i had been removed from the system.

Such an utility function is *aligned* with the global utility; in fact, since the second term in equation 2 does not depend on the action taken by agent a_i, any action that improves $DU_i(x)$ also improves the global utility $G(x)$. Furthermore, it is more *learnable* than TGU because, by removing agent a_i from the dynamics of the system, it provides a clearer signal to agent a_i.

In the case of intersection management, the driver agent computes the $DU_i(x)$ as follows:

$$DU_i(x) = (1 + P(x)) \cdot D(x) - (1 + P(CL_i(x))) \cdot D(CL_i(x)))$$

where $CL_i(x) = \langle x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \rangle$

3.3 Probability Collectives (PC)

Once the agents in a collective have been provided with "well-designed" private utility functions, many methods are available for supporting the agent decision making, such as reinforcement learning [5]. In this paper we draw upon a novel method called Probability Collectives (PC) [10], which has been developed within the COIN [7, 8, 9] framework, for the agent decision making. PC replaces the search for the most valuable action across the space of *actions* with the search across the space of *probability distributions* over those actions. In other words, PC aims at learning the agent decision strategies that maximize the global objective.

Formally, let $\mathcal{C} = \{a_1, a_2, \ldots, a_n\}$ be a collective of n agents. Each agent a_i can take an action by setting its action variable x_i, which can take on finite number of values from the set X_i. So these $|X_i|$ possible values constitute the action space of the agent a_i. The variable of the joint set of n agents describing the collective action is $x = \{x_1, x_2, \ldots, x_n\} \in X$, with $X = X_1 \times X_2 \times ... \times X_n$.

Given that each agent has a probability distribution (i.e. mixed strategy in game theory sense) over its possible actions, $q_i(x_i)$, the goal of PC is to induce a product distribution $q = \prod_i q_i(x_i)$ that is highly peaked around the x that maximize the objective function of the problem, and then obtaining the optimized solution x by sampling q.

The main result of PC is that the best estimation of the distribution q_i that generates the highest expected utility values is the minimizer[3] of the Maxent Lagrangian (one for each agent):

$$\mathcal{L}_i(q_i) = E_q[g_i(x)] - T \cdot S(q_i) \quad (3)$$

where q_i is the agent probability distribution over the agent a_i actions, x_i; $g_i(x)$ is the agent a_i private utility function (e.g. the *Difference Utility* defined in equation 2), which maps a joint action into the real numbers; the term $E_q[g_i(x)]$ is the expected utility value for agent a_i, subjected to its action and the actions of all the agents other than a_i; $S(q_i)$ is the Shannon entropy associated with the distribution q_i, $S(q_i) = -\sum_{x_i} q_i(x_i) \ln[q_i(x_i)]$; T is an inverse Lagrangian multiplier, which can be treated as a "temperature": high temperature implies high uncertainty, i.e. *exploration*, while low temperature implies low uncertainty, i.e *exploitation*.

Since the Maxent Lagrangian is a real valued function of a real valued vector, it is possible to use gradient descent or Newton methods for its minimization. Using Newton methods, the following update rule is obtained:

$$q_i^{t+1} = q_i^t - \alpha q_i^t \times \{\frac{E_q[g_i|x_i] - E_q[g_i]}{T} + S(q_i^t) + \ln[q_i^t]\} \quad (4)$$

where $E_q[g_i]$ is the expected utility, $E_q[g_i|x_i]$ is the expected utility associated with each of the agent a_i's possible actions, and α is the update step. Equation 4 shows how the agents should modify their distributions in order to jointly implement a step in the steepest descent direction of the Maxent Lagrangian.

Since at any time step t, an agent might not know the other agents' distributions, in this case it wouldn't be able to evaluate any expected value of g_i, because they depend on the full probability distribution q. Those expectation values can be estimated by repeated Monte Carlo sampling of the distribution q to produce a set of $(x; g_i(x))$ pairs. Each agent a_i then uses these pairs to estimate the values $E_q[g_i|x_i]$, for example by uniform averaging of the g_i values in the samples associated with each possible action.

3.4 PC for intersection management

PC is a broad framework for the analysis, control and optimization of distributed systems that offers new approaches to problems. Nevertheless, in order to be actually instantiated in a particular domain, several design decisions must be made.

Since the entire framework is based on the Monte Carlo-based estimation of the product distribution that maximizes the global objective, it is necessary to have a communication structure that enables to build the set of sampled joint actions. For example in [6] such a set is constructed using a token-ring message passing architecture. In this work, we opted for letting the agents asynchronously request the other agents in the collective to sample their distributions. Then each agent constructs locally its set of sampled joint actions and uses them to update its distribution with equation 4. We assume that the agents truthfully sample their distributions without manipulation, even if investigating how an agent can exploit the coordination mechanism for its purposes deserves a further analysis.

Another design decision is the setting of the initial temperature T and the initial probability distribution q_i. The initial temperature usually depends on the particular domain, because its order of magnitude is strictly related with the expected utility values (see equation 4). In our experiments we set the initial temperature to 1. On the other hand, the initial probability distribution q_i is usually initialized with the maximum entropy distribution, i.e. the uniform distribution over the action space X_i. In this way we don't make any assumptions about the desiderability of a particular action and all the actions are equiprobable.

[3] Without loss of generality, the global utility function is considered as a "cost" to be minimized, by simply flipping the sign of the utility value

Usually, the Lagrangian minimization proceeds as follows: for a given temperature T, the agents jointly implement a step in the steepest descent direction of the Maxent Lagrangian using equation 4. Then the temperature is slightly reduced, and the process continues, until a minimum temperature is reached. The annealing schedule we implemented was geometrically reducing the temperature T as long as a driver agent approaches the point after which it is obliged to send a request to the intersection manager, as described in section 2.1. When a driver agent arrives at that point, it evaluates the action with the highest probability, sets its velocity accordingly (see section 2.2) and makes a reservation request with the given velocity.

Algorithm 1 sketches the algorithmic structure of an agent program that implements PC for the intersection management problem. The algorithm starts initializing the temperature T and the probability distribution q_i (line 01 and 02). The main loop controls the annealing schedule of the temperature T (line 09), until the driver agents reaches the minimum distance to the intersection (line 03).

The minimization of $\mathcal{L}_i$ for a fixed temperature is accomplished by repeatedly determining all the conditional expected values $E_q[g_i|x_i]$ (line 06) and then using these values to update the distribution (line 07). Such values are obtained by requesting samples to the agents in the collective (line 04) and storing them when they are received (line 10), in order to have an estimation of the entire distribution q.

At the end of the algorithm, agent a_i selects its "best" action by sampling the distribution q_i or directly selecting the action with the highest probability, and then store the request that will be sent to the intersection manager.

From this point on, the agent starts to behave like in the reservation-based scenario described in section 2 (for more details, see [2, 3]). It sends reservation requests to the intersection manager, until it receives a confirmation or a refuse message. In the first case, the driver agent stores the reservation details and tries to meet them. Otherwise, it decreases its velocity and makes another request in the next step.

A driver agent is not allowed to cross the intersection with an out-of-date reservation or without reservation at all. A confirmed reservation goes out-of-date if the agent is not able to meet its details, i.e. the agent cannot be at the intersection at the time specified in the reservation, due to the traffic conditions. In this case, the driver agent can cancel the reservation with the intersection manager and make a new one, whose constraints it is able to meet.

If an agent arrives at the intersection with no confirmed and valid reservation, it is obliged to stop at the intersection. At this point, the driver agent is only allowed to propose reservations for the time slots in the near future.

4. EXPERIMENTAL RESULTS

In this section we present the results of the experiments conducted with the simulator of a *4-ways-3-lanes* intersection (see figure 1). The metric we used to evaluate the efficiency of the intersection was the average travel time of a set of vehicles. During the simulation, a total of 100 vehicles are generated using a Poisson distribution $f(k, \lambda) = \lambda^k e^{-\lambda}/k!$, where λ is the number of expected occurences (i.e. vehicles) in a given interval. In all the experiments, the λ parameter is kept fixed, while we progressively reduce the interval, simulating in this way different (increasing) traffic densities

Algorithm 1 PC for intersection management

```
01: T ← 1
02: q_i ← uniformDistribution
03: while minimum distance not reached do
04:    requestMCsamples
05:    if m not empty then
06:       ce ← evalConditionalExpectations(m)
07:       q_i ← updateQ(ce)
08:    end if
09:    T ← updateT
10:    m ← storeIncomingMCSamples
11: end while
12:
13: x̂_i ← mostProbableAction
14: velocity ← x̂_i.arrivalVelocityAtIntersection
15: store request R = ⟨vehicleID, direction, lane,
    turn, arrivalTimeAtIntersection, velocity⟩
```

Each spawned vehicle has a preferred velocity, whose value is generated randomly using a gaussian distribution with $\mu = 3$ and $\sigma^2 = 1$, and the maximum allowed velocity was set to 10.

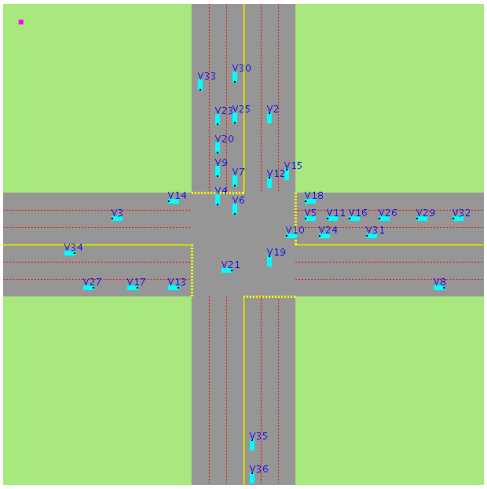

Figure 1: Simulator snapshot

One challenge in the implementation of the coordination mechanism was coping with the extreme dynamic and asynchronous nature of the system, as well as with the constraints imposed by the real-time.

Furthermore, while in multiagent reinforcement learning it is assumed that in every learning episode the set of agents remains the same, in this case this assumption does not hold, because the set of learning agents is created dynamically. Once a driver agent appears in the managed area, its ID is stored by the road infrastructure. Then the road infrastrucuture periodically communicates the set of collected IDs to the agents, in order to create collective of coordinating agents.

In figure 2 we plotted the average travel time of 20 experiments for two different configurations. In one configuration, each driver agent communicates exclusively with the intersection manager by making reservation requests solely on the basis of its knowledge; in the other configuration, the

driver agents implement the coordination mechanism before starting making reservation requests.

If the traffic density is low, the average travel time of the two configurations is approximatively the same. This is reasonable, since with low traffic density few reservation requests are rejected, so no previous coordination is needed. Similarly, with high traffic density the average travel time tends to be the same for the two configurations. Again this is reasonable, because the intersection tends to be saturated by vehicles stopped at the intersection, waiting for its reservation request to be confirmed.

On the other hand, it is noticeable a traffic density interval where the coordination between driver agents benefits the intersection system by reducing the average travel time up to approximatively the 7% (see figure 3).

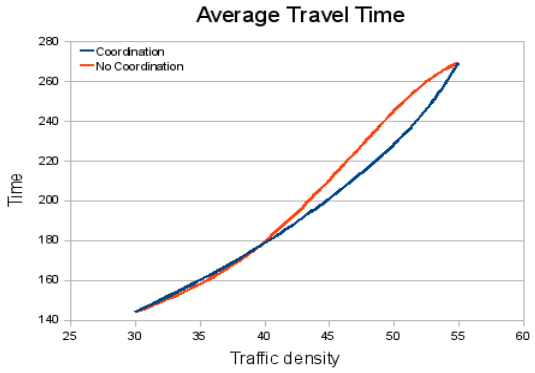

Figure 2: Average travel time

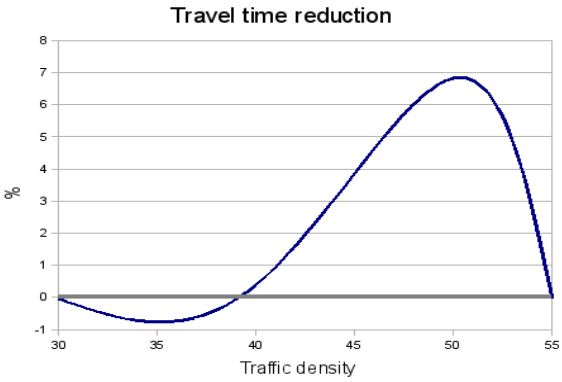

Figure 3: Average travel time reduction

Notwithstanding, there is space for further possible improvements of the agents learning capabilities. Firstly, the agent action space to act in the environment is quite reduced, since it can only set the velocity at which it intends to cross the intersection. For example, if there is a confirmed reservation of a very slow vehicle, which occupies the intersection for many time slots, it is reasonable to think that there is no way for an approaching agent to make a request that will not be refused, no matter the velocity it proposes. So, a possible improvement could derive from giving the agents the possibility of changing its lane.

Furthermore, with the agent model described in section 3, a collective of agent searches the product distribution q to maximize a global utility function $G(x)$. This is a function of the joint action x, and do not take in consideration external factors (i.e. noise). In the domain of the intersection management, for a given $x = \langle x_1, x_2, \ldots, x_n \rangle$, an agent is only able to evaluate the number of conflicts that occurs among the x_is and their travel times, by simulating the journey of each agent a_i through the intersection. If for example the intersection is saturated due to a crash, or it has been reserved by very slow vehicles, the collective is not able to react to these events and adjust its collective behaviour, since it does not have such information.

A way to circumvent this problem is modifying the structure of the global utility as a function of a 2-players game between the collective and the external world. At each time step, the collective sets its joint action x, while the world plays y. Such a vector y contains any external information not directly under control of the collective. Then the global objective $G(z)$ is calculated, as a function of the full vector $z = \langle x, y \rangle$. In the domain of the intersection management, the vector y could contain information about the traffic conditions in front of each driver agent, or about the confirmed reservations that the intersection manager has in its database. Nevertheless, such modification could have side effects that may worsen the learning rather than improving it.

Another issue that it is worth mentioning is that the multiagent learning performed by the driver agents comes as a sort of coordination on-the-fly: an agent does not learn from the behaviours of the other agents that it observes, rather such information is explicitly provided by exchanging samples of the agents' distributions. This setting speeds up the learning, although it has an associated cost for the communication overhead. Basically the key point for an agent is evaluating the expected utility of its actions, $E_q[g_i|x_i]$ (see equation 4). Within this formalization, such a value is obtained using an estimation of the joint distribution q, by using the samples provided by all the agents.

If we remove the communication between agents, the only way for an agent to evaluate the expected utility $E_q[g_i|x_i]$ is actually executing different actions in several episodes (i.e. crossing the intersection several times at different velocities), then using the utility values of the different actions it has executed in these episodes to compute $E_q[g_i|x_i]$ and adapt its mixed strategy q_i. Within this formalization, no communication is needed, although the learning process will take much more time.

5. CONCLUSIONS

This paper showed that the intersection management problem offers many opportunities for multiagent learning [4]. In particular, we started from the COIN framework for the definition of agent private utilities, and we applied Probability Collectives to make the agents learn to coordinate their action. The preliminary experiments showed some improvements of the intersection efficiency, with a reduction of the average travel time for a given traffic density interval.

Future works includes evaluating the model under different metrics (e.g. delay, congestion, number of refused reservation...), considering different private utility functions and global objectives, as well as modifying the model as described in section 4. More generally, the road traffic management scenario is open to a plethora of interesting research lines, from the study of "cooperative vs competitive" agent

behaviour, to the impact of "malicious" agents that try to exploit the coordination mechanism.

6. REFERENCES

[1] F. Klügl, A. Bazzan, and S. Ossowski. Applications of Agent Technology in Traffic and Transportation. Birkhäuser, 2005.

[2] K. Dresner, and P. Stone. Multiagent traffic management: A reservation-based intersection control mechanism. *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 530–537. ACM Press, 2004.

[3] K. Dresner, and P. Stone. Multiagent traffic management: an improved intersection control mechanism. *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 471–477. ACM Press, 2005.

[4] K. Dresner, and P. Stone. Multiagent Traffic Management: Opportunities for Multiagent Learning. Lecture Notes in Computer Science, pages 129–138, Volume 3898, Springer-Verlag, 2006.

[5] R.S. Sutton, and A.G. Barto. Reinforcement learning: An Introduction. MIT Press, 1998.

[6] A. Waldock, and D. Nicholson. Cooperative Decentralised Data Fusion Using Probability Collectives First International Workshop on Agent Technology for Sensor Networks (ATSN-07), held at AAMAS 2007

[7] Wolpert, D., and Tumer, K.: *An introduction to COllective INtelligence*. Technical Report NASA-ARC-IC-99-63, NASA Ames Research Center, 1999

[8] Wolpert, D., Wheeler, K. R., and Tumer, K.: *General principles of learning-based multi-agent systems*. In O. Etzioni, J. P. Müller, and J. M. Bradshaw, editors, Proceedings of the Third Annual Conference on Autonomous Agents (AGENTS-99), pp 77–83. ACM Press, New York, 1999

[9] Wolpert, D., and Tumer, K.: *Optimal payoff functions for members of collectives*. Advances in Complex Systems, 2001

[10] D. Wolpert. Information theory - the bridge connecting bounded rational game theory and statistical physics. ArXiv Condensed Matter e-prints, 2004.