2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)
Windsor Oceanico Hotel, Rio de Janeiro, Brazil, November 1-4, 2016

A Distributed, Collective Intelligence Framework for Collision-Free Navigation through Busy Intersections

Rahi Kalantari, Michael Motro, Joydeep Ghosh, and Chandra Bhat

***Abstract*—Connected autonomous vehicles have the potential to cooperate at a higher level than human drivers, for instance to navigate busy intersections with relaxed traffic rules. This paper presents a novel system for autonomous traffic management through intersections based on distributed cooperative intelligence. In this framework each vehicle decides its own motion, rather than entrusting control of all vehicles to a central agent which then becomes a single point of failure. The motion plans are developed by reinforcement learning on simulations of intersection traffic. The developed system is shown to route vehicles to their destination in a safe and timely manner. In the near future, this system can be deployed to as a collision warning/collision avoidance aid to human drivers by selectively acting on the discrepancy between a human's behavior and the AI-determined safe behavior.**

I. INTRODUCTION

In the not-so-distant future, cars may become autonomous entities with the capability to make decisions at the individual-vehicle level to meet a particular passenger's needs. However, for optimal performance at the level of entire transportation systems, such autonomous cars will need to work collectively, collaborating so that all the vehicles in a network safely reach their destinations in the shortest possible time. This paper presents a distributed artificial intelligence (AI) framework to strive towards this goal. It assumes that vehicles can gather adequate knowledge of their environment, for instance through V2V communication, but make individual, fully autonomous decisions. In such scenarios, how can each vehicle make decisions that not only benefit its own goal but also simultaneously optimize a long-term "global" utility computed over the entire transportation system?

We focus on the specific task of intersection navigation. Intersections, especially in large urban centers, are areas of high risk for all drivers. According to data released by Fatality Analysis Reporting System in 2009, about 40.1 percent of all crashes in the United States were related to intersections [16]. Traffic lights at these intersections are programmed to control and route traffic safely, at the cost of speed and convenience for each driver. In this paper, we consider how automobiles could be designed to communicate with one another and learn the appropriate time to execute an intersectional shift, judge the speed at which to cross, and comprehend the proximal distance of other vehicles without the need of human drivers.

* Rahi Kalantari, Michael Motro, and Joydeep Ghosh are with the Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78710 USA. Phone: +1(512) 888-3195, email: rahikalantari@gmail.com.
Chandra Bhat is with Civil, Architectural, and Environmental Engineering Department, University of Texas at Austin.

It is common in large multi-agent systems to design a global utility function, where each agent has its own private utility function but the ultimate objective of the system is to optimize the global utility. Note that it is easy for an agent to learn to optimize its personal utility; the crucial problem is how to design mechanisms that derive benefit from the personal utility functions of the agents so that they "cooperate unintentionally" and optimize the global utility.

Collective Intelligence (COIN) [4] is a method for a large system of agents to learn a set of actions in specific situations that maximize the individual private utility functions while simultaneously maximizing global utility, without any centralized communication or control among the agents. In this work COIN is adapted to route automated cars safely and efficiently through an intersection.

II. RELATED WORKS

There is a wide range of techniques of collision avoidance. In some methods, machine learning techniques have been utilized while some other use a variety of other approaches from optimization to simple rule based methods. Guy and Chungani in [18] find a collision free velocity for each agent by constructing the cone of potentially colliding velocities, and the optimum velocity is then found at the boundary of union cones between the target agent and all other agents. Beside the approaches based on the velocity obstacles, other methods proposed for collision avoidance and navigation among non-static agents include [1], [2] and [3]. These works do not fully take into account the fact that the motion of some obstacles, namely other agents (vehicles), can be affected by the presence and motion of a given agent. Also, these methods are more suitable in robotic domains as agents does not need to follow certain paths and have almost full degree of freedom in a 2-D grid, in contrast to constraints imposed by road lanes, driving rules, etc, that autonomous vehicles need to face.

Strategies to automate vehicles are typically categorized by specific maneuvers such as car-following, lane-changing, and overtaking. Car-following, also known as cooperative adaptive cruise control, has been heavily researched and includes reinforcement learning [8]. Choosing lanes on a freeway was framed as a cooperative, multi-agent problem by [19], and since has been undertaken with several variants of reinforcement learning. [9] and [10] in particular use distributed Q-learning methods much like COIN, but their problem formulation is quite different as lane-changing is the only vehicle action made autonomous.

Both methods were shown to outperform rule-based algorithms, in terms of the average speed of vehicles.

Cooperation between vehicles is especially necessary at intersections. Information exchange among vehicles can provide reliability and increased efficiency by relaxing the constraints on when vehicles can navigate the intersection. [15] surveys the many types of automated or connected intersection management.

Most cooperative intersection management methods first identify the regions in which vehicles on two different paths can collide. The goal is to move vehicles through the intersection at an optimal rate, while ensuring that no two vehicles will simultaneously occupy a collision region. A straightforward approach is to hand over the command of all vehicle's actions to a central computer, most likely one installed at the intersection. This central agent can use optimization techniques to choose the best combination of actions. However, it is generally considered unreliable for a single unit to control all vehicles, and the computational requirements of finding the optimal motion for all vehicles at once are quite high. An alternative approach is to use the central agent to dictate certain aspects of the vehicle's action, via a set protocol, while the rest of the action is left to the driving agent. The most common type of protocol is centralized reservation, wherein each vehicle must ask the central agent for permission to enter the collision region. The central agent's responsibility is to ensure that no vehicles enter these regions simultaneously. [11] and [12], for instance, present centralized reservation systems and show them to be highly efficient. Such solutions require each vehicle to follow the reservation system of any particular road, and they cannot be easily adapted as mere aids to human-driven cars.

If a method of uniquely reserving space in the intersection is constructed carefully, it is possible to forego the central agent and instead perform distributed reservation through vehicle-to-vehicle (V2V) communication. [14] develops a distributed reservation system and demonstrates that it can be used for research. The few other works that perform distributed intersection management with only basic broadcasts also rely on some form of common knowledge between vehicles. [7] develops a ranking technique that always prioritizes one out of any two conflicting vehicles. [13] develops a more complex, but similar, method based on a schedule tree. No known works have tackled decentralized intersection management with learning-based methods. While this paper does not use COIN on an unsignalized intersection, the same framework could easily be applied for such a task. The testing environment in this paper is a signalized but unprotected intersection, so vehicles turning left or right will still need to cooperate with conflicting traffic.

III. PROBLEM FORMULATION

Our agents in this framework are fully autonomous cars (one agent per vehicle) that know their location and destinations and communicate through DSRC with all nearby vehicles, gathering their approximate location and speed. These agents are trained in a simulation environment. The goal of each agent is to reach their destination in a timely manner while avoiding accidents, keep a safe distance to the car ahead of them, and traverse the intersection without causing issues for the other cars in the environment. COIN addresses this optimization problem by decomposing it into many sub-problems, each of which can be solved individually (car level instead of intersection level) and which are formulated so that the Nash equilibrium of all agents will optimize the Global Utility [4]. Using this approach does not mean that those problems are independent of one another, but that each individual solution is a contributor to solving the system-level problem. The collective intelligence formulation of this problem uses a difference reward, so that maximizing private utility functions is equivalent to finding Nash equilibria of all the agents for their state-action space, resulting in optimization of the global utility [4].

A. System Evaluation

This subsection describes the global utility components, which individually focus on avoiding collisions, directing vehicles to their destination if it is safe to do so, keeping safe distance to the vehicle ahead, reducing the number of unnecessary maneuvers, and doing all these in a timely manner. The global utility function, G(z), is the sum of these terms. Specifically, G(z) is expressed by its components as follows, here S is set of all agents and z represents tuples of $(z_1, z_2, \ldots, z_{|s|})$ where Z_i is the i^{th} agent's State-Action space:

1. Speed reward measure:

$$D_j(Z) = 2(S(Z_{g_j}) - S(Z_g^{\mathrm{Max}})\,)/\,S(Z_g^{\mathrm{Max}})$$

where $S(Z_{g_j})$ is the scaled speed of car j at the instance of t and $S(Z_g^{\mathrm{Max}})$ is the max scaled speed allowed on that road.

2. Following the correct path:

$F_j(Z) = 0$ if the vehicle is in a road/lane that will directly lead to their destination, and -10 otherwise.

3. Proximity to the closest vehicle ahead of this vehicle:

$$P_j(Z) = h\left(\min_{i \in X_j} TTC(Z_j, Z_i), \min_{i \in X_j} DTC(Z_j, Z_i)\right)$$

X_j is the set of all vehicles that cross the intersection at apprixmatly same time as vehicle j, as well as vehicles in the same lane as and ahead of vehicle j. Vehicles approaching from behind or from adjacent lanes are not considered for this reward, so that responsibility for collision avoidance while car-following or lane-changing is primarily deferred to the car making the action.

$TTC(Z_j, Z_i)$ is the number of seconds in the future at which vehicles j and i will collide, if they maintain their current paths and velocities. If the

vehicles never collide, *TTC* is set to a high positive number.

$DTC(Z_j, Z_i)$ is the distance between vehicles *j* and *i* that will collide, if vehicle *i* is directly ahead of vheicle *j*, and otherwise a high positive number.

h(TTC,DTC) is a function that discretizes the reward based on the time and distance to collision.

4. Collision Reward:

   $C_j(Z)$ = -1500 if vehicle *j* collided with another vehicle at time t, and 0 otherwise.

5. Unnecessary lane change manoeuvre:

   $L_j(Z)$ = -20 if a lane change was made that does not put vehicle *j* on the correct path to their destination.

6. Turn decision:

   $T_j(Z)$ = 0 if vehicle *j* makes the correct turn choice at the intersection, and -75 otherwise. By its nature this reward only applies at one point in time (for each vehicle).

D(z) provides the proximity of the running car to the maximum speed allowed on the road at each instance of time and charges the agent with the fractional amount of deviation from the max speed at that instance. *F(z)* negatively rewards the cars that are not in the path to their predetermined destinations. *P(z)* decreases reward of agents that choose to get close to the other cars, this reward has two components one is calculated based on the approximate time that car trajectories may cross in the future if both of the vehicle drive with the same speed, and the other component is the relative distance of vehicle with respect to their speed from each other, then the first is related to approximate time to hit the cars passing across the intersection and the other is the component that approximate the distance relative to own speed to the car ahead or ones that cross the intersection. *C(z)* is a one-time heavy negative reward that the agent is charged at the time of collision. *T(z)* is one-time negative reward for making wrong decision turn at the intersection. *L(z)* is a negative reward for vehicles which make lane changes that place them in an incorrect path to their destination.

Note that each outcome is penalized differently based on its seriousness, with collisions having a much higher penalty than any other event. Having defined the components of the local utility above, the local and global utility functions are:

$$G_j(z) = D_j(z) + F_j(z) + P_j(z) + C_j(z) + L_j(z) + T_j(z)$$

$$G(z) = \sum_{j \in S} G_j(z)$$

This is the global utility function for a single point in time. COIN maximizes the global utility function across $(z_1, z_2, \ldots, z_{|s|})$ tuple at any time instant.

B. Agent-Based Intersection Traffic Management

The multi-agent approach that we present is designed to direct agents evolving independent solutions that maximize the system-level ("global") evaluation function. Encompassing a realistic driving environment, and ensuring that the agents can be trained with reasonable computational resources and time, are also design considerations. This section describes significant elements of our system designed based on COIN framework.

State Space:

One of the biggest challenges for COIN is to construct a state space that is not so large as to make the learning task intractable, but still always includes the necessary information to make a correct decision. One natural choice of the state space is the detailed location of each agent on the road and their speed (agents' trajectories), but it is easy to see that this agent space is far too large and reinforcement learning could get intractable. Our proposed state-space addresses this issue.

This list includes every element of the states that the agents learn on: note that all variables are ordinal or categorical, and some are heavily rounded to keep the state space small.

Location = {pre-intersection, entering intersection, within intersection, post intersection}, this specifies the rough position of the car in the intersection.

Lane ={Agent's current lane number depending on road }

Lane to Destination = {True, False} whether the path that agent is currently in ends up in its desired destination (e.g., the car that aims to go straight should learn not to take the lane that is specifically meant to be used by vehicles turning left.)

Adjacent Lane is Free = {None, Left, Right, Both}, measures the presence of other vehicles in adjacent lanes

Agent Speed = {0, 10, 20, 30, 40, 50, 60} (Km/h), Agent speed with resolution of 10 Km/h

Time to Nearest Vehicle = {[0,2],[2,4],[4,8],[8,∞]} (seconds), the shortest time-to-collision with any other vehicle ahead or crossing the agent trajectory, using a constant-velocity model to predict vehicles' trajectories. As discussed in the *System Evaluation* section, this is performed by first calculating the positions along each path through the intersection where the vehicles could collide, then checking for vehicles that will simultaneously occupy conflicting paths. For all features involving time or distance between other cars, if there is no relevant car then the value is set at the highest level.

Slower Time to Nearest Vehicle = {[0,2],[2,∞]} (seconds), checks for potential collisions under the assumption that this vehicle moves 10 Kph slower than its current speed.

Faster Time to Nearest Vehicle = {[0,2],[2,∞]} (seconds), checks for potential collisions under the assumption that this vehicle moves 10 Kph faster than its current speed.

Distance to the car ahead = {[0,2],[2,4],[4,∞]} (meters), the approximate distance to the car ahead (on the same path)

Distance to a crossing car = {[0,2],[2,4],[4,∞]} (meters), the approximate distance to a car that is blocking this car's way through the intersection.

Priority = {True, False}, this state is designed to address four ways traffic at the intersection. This could be thought as a synchronous timer between the agents which drive on one of the two perpendicular roads that create the intersection, similar to a traffic signal. Further details and motivation behind this state and action to be picked at that state are explained in *Intersection right-of-way* section.

Agent Action:

An agent's action at each step consists of changing speed, changing lane, and turning (at the intersection entrance only).

- Speed={+10, -10, 0} (Km/h)
- Lane Change={Left, Right, None}and
- Turn={Left, Right, Straight}

The full set of actions is not available in all Locations: in *pre-intersection* and *post-intersection* speed change and lane change are available actions, when *entering intersection* turn and speed change are available actions, and *within intersection* only speed change is available.

Agent Reward Structure Selection:

We used the difference reward function as each agent's private utility function. The difference reward is defined as:

$$R_i = G(z) - G(z - z_i + c_i) \quad (1)$$

where z_i is the state space-action of agent i. Here z contains all components of all agents' state and action at each time instance, and those z components that are affected by agent i (z_i) are replaced with the fixed constant c_i. In this case, the simplest way to define c_i is by taking the agent i "off the grid," and re-evaluating the global reward without the presence of agent i. There are two advantages to using a difference reward structure. The first effect is on learnability, which is the measure of sensitivity of an agent's private utility function with respect to change of its state. The second term in Eq.(1) removes a significant portion of the impact of other agents in the system which makes the evaluation function more sensitive to i^{th} agent action. Secondly, it affects the "factoredness" which is defined as the degree with which an agent's personal utility is aligned with the world utility. Since the second term does not depend on the actions of agent i, any action by agent i that improves R_i also improves G.

Agent Evolution Algorithm Selection:

The objective is that each agent learns to make decisions at any given state, that will lead to the best system evaluation G for that specific state. Here we assume that each agent will have a reward function (as described in the previous section) and will aim to maximize its reward using its own reinforcement learner. For complex delayed-reward problems, more sophisticated reinforcement learning systems such as temporal difference may have to be used. However, due to our agent selection, state space design, and agent action set, we have picked simple reinforcement learning techniques to utilize immediate rewards. As a consequence, a simple table-based immediate reward reinforcement learning approach is used. Our reinforcement learner is equivalent to an α-greedy Q-learner with a discount rate of γ=1.

At every time instance an agent takes an action and then receives a reward after evaluating that action, and the Q table gets updated to represent the value of taking that action in that state, as follows:

$$Q_0(s,a) = (1-\lambda)Q(s,a) + \lambda\left(R(s^+) + \max_{a^+} Q(s^+,a^+)\right)$$

λ is the learning rate

s^+ is the next state, i.e. the state that was reached by performing action a

At every time-step the agent chooses the action with the highest table value with probability $1 - \alpha$ and chooses a random action with probability α. In the experiments described in this paper, α is equal to 0.33. λ is equal to $\lambda = \frac{1}{n(s,a)^{0.5+\delta}}$, where $n(s,a)$ is the number of observed instance of that state-action point during the training time and δ is some small positive constant. The parameters were chosen experimentally, and we note that system performance was not overly sensitive to these parameters.

To sufficiently train every state-action combination would take extremely intensive simulations, but the common or pertinent states are trained rapidly. To deal with occasional rare states whose actions have not all been trained (during testing), KNN technique for functional approximation has been utilized. We observe the optimal actions of similar states and choose the most commonly selected action.

C. Intersection right-of-way

A common challenge in multi-agent systems is that all agents learn to act the same way, even though in some scenarios it is necessary for two agents in similar situations to each choose a different action. Especially in this case that state-space does not offer the luxury of knowledge to the exact location of the agents on the road and states may end up to be mirrored for all cars that enter to the intersection from four different sides of the intersection. For the task of autonomous navigation, there is the possibility that two or more vehicles may detect a potential collision with the other(s) and stop immediately. This is especially problematic if each vehicle stops while in the

intersection, preventing other vehicles from progressing, as shown in Figure 1.

This issue was addressed by assigning each road a priority, which varied over time. The rewards each vehicle receives are adjusted based on this priority. If a vehicle with higher priority and a vehicle with lower priority collide, then the higher priority vehicle is punished less heavily than the lower priority vehicle.

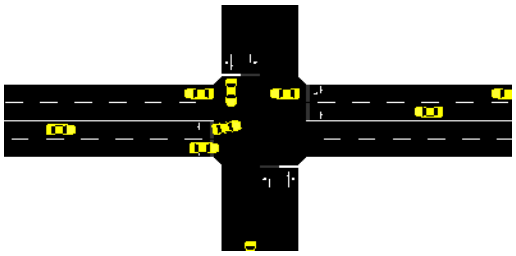

Figure 1: Example of intersection deadlock between three vehicles

IV. RESULTS OF COLLECTIVE INTELLIGENCE TRAINING AND SIMULATIONS

In this section, we describe some specific scenarios for training and testing provide statistical results obtained during testing. The simulated driving environment is a symmetrical four-way intersection. First, the training was performed for a specific scenario with a constant number of cars, fixed time spacing between the cars, and fixed starting speed. The scenario was designed so that every pair of cars mutually meet at the intersection and learn how to avoid a collision as they proceed to their destination. The training converged quickly for this specific scenario. Then, training and testing were completed for cars making left turns at the intersection against oncoming traffic use randomized starting times and speeds for all vehicles., then it was expanded to the two-way opposite directional incoming traffic with arbitrary destination point.

The four-way incoming traffic scenario using the techniques, which have been described in *Intersection right-of-way* section, is a natural expansion of the two-way opposite-directional incoming traffic with one simple rule based technique to force the agents without priority to stop at the intersection and the ones with priority to proceed and flow through the intersection. The system was further trained to capture configurations that were not explored in two-way training. These corner points can be interaction of leftover vehicle at the intersection from one 2-way traffic direction (e.g. North-South) and new 2-ways opposite traffic direction (e.g. East-West) which are entering the intersection. Having much more training episodes, still left the system with many untrained state-actions in testing that encouraged us to incorporate functional approximation.

The best way to test the learning capabilities of this system is to see if it picks up the correct timing of a given maneuver. For unprotected left turns, after less than 100 simulations, vehicles learn to wait when their path would cause a collision with an oncoming vehicle, as well as to speed up or slowdown in response to the other vehicles taking the same route. The Figure 2 demonstrates the second scenario where the autonomous agents learnt how to manage the traffic in the road and intersection to fulfill the requirements defined in previous section that essentially results in to safe travel.

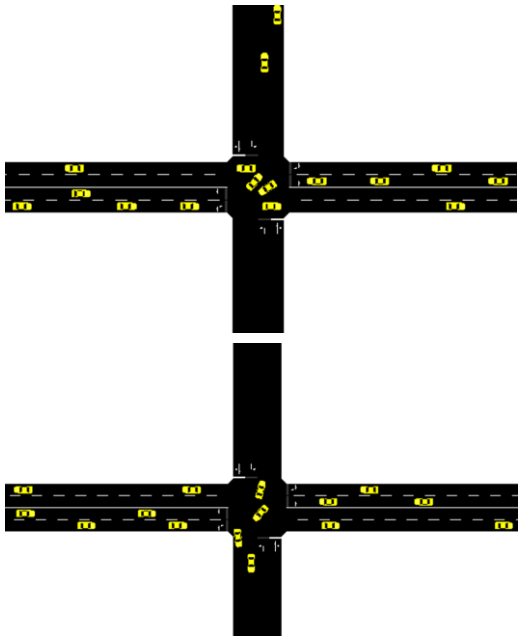

Figure 2: Two sequential snapshots of testing for bi-directional incoming traffic.

The testing results are presented in Table 1. Testing results show a learned model in action, unlike training results, which are gathered while the model learns and thus should clearly perform less well. Without the existence of competing baseline algorithms, the training results provide the simplest baseline to check against. On two-way traffic, the system achieves less than 1% collision rate during testing. By comparison, the number of collisions during training exceeds 25%. Next, training was completed for the full four-way intersection with a random number of cars, random starting time, random starting speed, and random destination. Under this scenario, there was about 29% collision during the training while collision during testing was about 5%. The system also showed relative robustness in face of the white noise added to the locations of all agents at any point of time. The noise has mean of zero and standard deviation of 1m. The results can be observed in Tables I that our techniques tested under such noise effect.

Live demos are available to showcase the capabilities of the system under a variety of intersection scenarios.

TABLE I: RESULTS FOR Q-LEARNING

Maneuver	Average number of cars per episode	Collision (%)	Wrong path (%)	Average time (seconds)	Number of random episodes for testing
2-ways	45	0.65	2.5	14.7	20
4-ways	85	5.3	12.6	12.2	15
2-ways +wn	45	2.7	3.8	14.1	10
4-ways +wn	85	7.5	15.0	18.1	10

V. FUTURE WORK

A natural expansion to this work is to determine actions for untrained states using the correlated trained states and pattern of the training matrix with sparse recovery techniques as an approximation function. A second possible expansion to this work could be done by learning how to drive to intersection from all direction without the *soft-light* using temporal difference reinforcement learning. In parallel, we can derive a more suitable state space using deep learning techniques for this problem similar to [5] or [6].

VI. CONCLUSION

This paper introduced a novel approach to distributed, autonomous navigation through traffic intersections based on a reinforcement learning oriented collective intelligence framework. In simple situations, the method yielded efficient and safe driving strategies. For the more complex task of navigating crowded four-way intersections, the strategy learned was not fully safe or time efficient, but the few cases of accidents should be resolvable by further exploration of the design space, leading to a unique and reliable solution to a key problem facing the design and deployment of autonomous, smart vehicles.

ACKNOWLEDGEMENT

This work was supported by Texas Department of Transportation under Project 0-6877 entitled "Communications and Radar-Supported Transportation Operations and Planning (CAR-STOP)."

REFERENCES

[1] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[2] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 1610–1616.

[3] J. van den Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2006, pp. 2366–2371

[4] Wolpert, David. "Theory of collective intelligence." In *Collectives and the design of complex systems*, pp. 43-106. Springer New York, 2004.

[5] Lange, Sascha, and Martin Riedmiller. "Deep auto-encoder neural networks in reinforcement learning." *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE, 2010

[6] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015): 529-533.

[7] G. Lu, L. Li, Y. Wang, R. Zhang, Z. Bao, and H. Chen. "A rule based control algorithm of connected vehicles in uncontrolled intersection." In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on* (pp. 115-120). IEEE, 2014.

[8] C. Desjardins and B. Chaib-draa. "Cooperative adaptive cruise control: A reinforcement learning approach." *Intelligent Transportation Systems, IEEE Transactions on*, 12(4), 1248-1260, 2011.

[9] M. D. Pendrith. "Distributed reinforcement learning for a traffic engineering application." *In Proceedings of the fourth international conference on Autonomous agents* (pp. 404-411). ACM, 2000.

[10] J. Laumonier and B. Chaib-draa. "Partial local friend multiagent learning: Application to team automobile coordination problem." *In Advances in Artificial Intelligence* (pp. 359-370). Springer Berlin Heidelberg.

[11] K. Dresner and P. Stone. "A multiagent approach to autonomous intersection management." *Journal of artificial intelligence research*, 591-656.

[12] J. Lee, and B. Park. "Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment." *Intelligent Transportation Systems, IEEE Transactions on*,13(1), 81-90, 2012.

[13] L. Li, and F. Y. Wang. "Cooperative driving at blind crossings using intervehicle communication." *Vehicular Technology, IEEE Transactions on*, 55(6), 1712-1724, 2006.

[14] R. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige. "Intersection management using vehicular networks." (No. 2012-01-0292). *SAE Technical Paper*, 2012.

[15] L. Chen and C. Englund. "Cooperative Intersection Management: A Survey." 2015.

[16] U.S. DOT National Highway Traffic Administration (NHTSA). "Traffic Safety Facts." http://www-nrd.nhtsa.dot.gov/Pubs/811402EE.pdf, 2009

[17] Guy, Stephen J., Jatin Chhugani, Changkyu Kim, Nadathur Satish, Ming Lin, Dinesh Manocha, and Pradeep Dubey. "Clearpath: highly parallel collision avoidance for multi-agent simulation." In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 177-187. ACM, 2009.

[18] Wolpert, David H., Kevin R. Wheeler, and Kagan Tumer. "General principles of learning-based multi-agent systems." In *Proceedings of the third annual conference on Autonomous Agents*, pp. 77-83. ACM, 1999.

[19] Moriarty, David E., and Pat Langley. "Learning cooperative lane selection strategies for highways." In *AAAI/IAAI*, pp. 684-691. 1998.