

Safely Managing Autonomous Intersections under Uncertainty

Abstract

While the stop-light system for managing intersections might work well with human drivers, autonomous vehicles that can communicate wirelessly with a central intersection manager allow for better routing through the intersection, saving energy and reducing delays. However, these vehicles will be unable to predict precisely when they will arrive at the intersection, resulting in the challenging task of scheduling under uncertainty. Further complicating the matter is that the intersection must schedule its use by many vehicles without explicitly controlling the vehicles themselves. Finally, saturation of this wireless network would degrade performance and safety, so communication between the intersection and the vehicles must be limited. We contribute an intersection management algorithm that handles uncertain predictions, guarantees safety, limits communication, and utilizes realistic assumptions about real-world autonomous vehicles. We show that previous algorithms that don't reason about uncertainty cannot achieve safe operation. The trade-off for reasoning about uncertainty is significantly increased computation, an issue which we assuage using probabilistic techniques to find an approximate solution. Finally, we substantiate our claims empirically on a small domain with real robots and a much larger domain in simulation.

INTRODUCTION

Motivation

Recent advances in artificial intelligence and robotics suggest that personal autonomous automobiles are soon to come. One important reason for the adoption of these vehicles is efficiency; people can get to their destinations faster, traffic congestion is decreased, the cost of the transportation infrastructure is lessened, and fuel is conserved. Intersections are an important piece of the traffic puzzle. Prior work showed that the stop light-system currently in use in most intersections (for human drivers) is not efficient for autonomous vehicles with the ability to communicate wirelessly (Dresner & Stone a). Their algorithm, FCFS or 'First-come, First-served,' exhibited superior performance in a simulation environment where motion was accurately predicted.

However, with real autonomous vehicles, motion cannot be predicted precisely. Inaccuracies spring from surface irregularities, position and distance estimation, lack of information about the plans of other agents, and imperfect motion models. In addition to improving efficiency, we wish to *guarantee* a safe routing, a formidable challenge in the face of such uncertainty. Finally, communication must be limited, or saturation of the wireless network will cause dropped packets and increased latency, ultimately resulting in lower efficiency and the loss of safety.

We give an algorithm that ensures safety and handles uncertainty, while limiting communication. We show empirical results on a small domain with real robots and a much larger domain in simulation. To our knowledge, ours is the first working, autonomous, multi-robot intersection to be successfully exhibited with real robots. Furthermore, Our simulation differs from previous work (Dresner & Stone a), insofar as we include stochastic motion to better simulate real autonomous vehicles.

Related Work

The problem of autonomous intersection management as a multi-agent system (MAS) was introduced previously (Dresner & Stone a). The authors propose a grid-based reservation system whereby: 1) a vehicle *predicts* when it will be in the intersection, 2) the vehicle *requests* the relevant grid cells at the appropriate times, and 3) the intersection manager *grants* the request if and only if it does not conflict with a prior request by another vehicle. Vehicles are free to drive as they please until they enter the intersection, at which point they must conform to their reservations.¹ The name of this algorithm is 'First-come, First-served,' abbreviated FCFS, referring to the fact that a vehicle that makes a reservation earlier will have priority.

While the authors demonstrate in simulation that FCFS is an improvement over the stop-light system, there are inevitably several limitations of their pioneering algorithm that we care to improve in our work:

- **Uncertainty.** Real vehicles will not be able to predict their motion precisely. As a result, a vehicle might make reser-

¹Neither FCFS nor our algorithm considers 'rogue' vehicles that do not follow the protocol; even in the stop-light system, safety is not ensured if a vehicle does not obey the lights.

vations it cannot keep, requiring it to cancel and issue another request. In fact, due to the priority property of FCFS, the vehicle has an incentive to make a reservation as *early* as possible, when it is *most* uncertain about its predicted arrival time in the intersection.

- **Inefficiency.** A vehicle holding a reservation that it will later change precludes other vehicles from utilizing that slot. When the reservation is canceled, FCFS does not redistribute the slot to vehicles that already have other reservations. Although the slot may be filled by newly arriving vehicles, it is unlikely that they would be able to make it in time (otherwise, they'd have already requested the slot and been rejected).
- **Excessive communication.** Due to the often-changing reservations, the vehicles exchange messages with the intersection manager very often. In fact, it is in a vehicle's best interest to try to get a better reservation *at each timestep*.

Refinements to FCFS (Dresner & Stone b) address the lat-
termost problem by specifying that a vehicle should only cancel a reservation if it is physically impossible to make it on time. However, this approach exacerbates the inefficiency due to uncertainty, since a vehicle may be locked into a 'pessimistic' reservation when it later finds out it could make an earlier slot.

Instead, we propose an algorithm that makes *lazy reservations* (in the spirit of 'just-in-time' scheduling (Jozefowska 2007)), waiting to make a reservation until it is necessary to do so for safety. In this manner, vehicles make reservations when they have the *most* accurate predictions about future motion. Additionally, our algorithm *reasons about uncertainty* in order to guarantee safety and improve efficiency. We use the contingency plan approach of (Bekris & Kavraki 2007) to guarantee safety while maintaining efficiency (specifically, the contingency plan we assume is that a vehicle can stop if it is told to do so before *stopping distance* dictated by its maximum deceleration capabilities).

These extensions comprise important contributions, both theoretically and practically. Reasoning about uncertainty and guaranteeing safety are fundamental distinctions from previous approaches, which have instead imposed stringent assumptions on being able to perfectly observe state and predict future state. It has always been a question whether previous approaches would apply to the real world, since results had heretofore only been tested in simulation. In this paper, we *relax* the assumptions of prior work to better reflect the real world, provide an algorithm that achieves *safety* and *efficiency* when models are inexact, and substantiate our results on *real robots*. To our knowledge, ours is the first working, autonomous, multi-robot intersection to be successfully exhibited in the real world.

Autonomous vehicles are an increasingly well-studied problem, with more recent attention being given to urban settings. Using artificial intelligence to manage intersections has even seen real-world use, notably with the SCOOT system (P. Hunt & Winton 1981), but still relies on traditional signaling techniques. Fully decentralized approaches have received some attention (Naumann & Rasche 1997) (Rasche

et al. 1997), as well as more global approaches among multiple intersections (Roozmond 1996) (Bazzan 2005).

Other works that have addressed scheduling under uncertainty include fuzzy logic techniques (Muhuri & Shukla 2008) and mixed-integer linear programming (Jia & Ierapetritou). The former utilizes heuristic techniques and, as with any fuzzy technique, circumvents Bayes Rule; while the latter must solve an NP-hard optimization problem, requiring an unacceptable delay in the face of our real-time requirements.

FRAMEWORK

In this section, we explain how we model the world, define relevant terms, and describe the task of autonomous intersection management.

Modeling the World

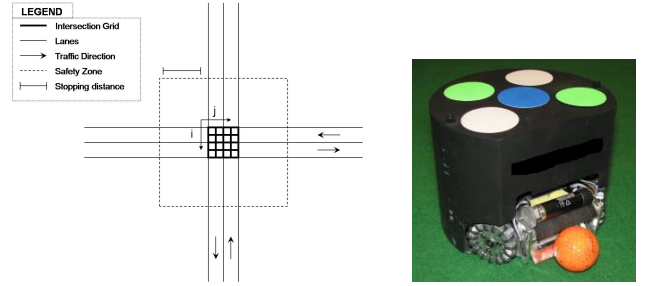


Figure 1: On the left, a conceptual diagram of the intersection; on the right, a small-size soccer robot, which we adapted for use with the intersection system.

- The *world* is the Cartesian plane \mathbb{R}^2 .
- A *lane* is an axis-aligned slice of the plane, such that the lane l is given by: $l = \{(x, y) | x \in [a, b]\}$ or $l = \{(x, y) | y \in [a, b]\}$.
- The *direction* a vehicle may proceed in a lane follows the right-hand rule of traffic (see Figure 1(a)).
- We divide the intersection into time-space *grid cells*. Since a vehicle has volume, it may occupy more than one grid cell at a given time. A grid cell address is given by the triplet (i, j, k) , where the last element is the timestep. Note the granularity of the grid is not necessarily the number of intersecting lanes, but can be more or less.
- A *route* is the trajectory of a vehicle through the grid over time, given as a set of grid cells: $\{(i_1, j_1, 1), (i_2, j_2, 2), \dots, (i_{t_{\text{exit}}}, j_{t_{\text{exit}}}, t_{\text{exit}})\}$.
- The *exit time* t_{exit} of a route is the time the vehicle leaves the intersection. Similarly, the *entry time* t_{entry} is when the vehicle enters the intersection.

Figure 1(a) illustrates an intersection labeled with these terms.

Modeling Uncertainty

- A *trajectory distribution* ρ of a vehicle v , generated at time t (using the information available to the vehicle at that time), is a probability distribution over the possible trajectories of the vehicle through a window $[t, T]$ of the time-space grid: $\rho(v, t) = \{\Pr(v \text{ occupies } (i, j, k))\}_{\forall (i, j, k), t \leq k \leq T}$. Note that we use the term *belief* interchangeably with trajectory distribution.

In our setting, uncertainty stems from stochasticity in motion, and results in route predictions that may or may not be realized. Abstracting away further details, a single vehicle's prediction task is to determine whether or not it will occupy a certain grid cell (or, more precisely, the probability it will occupy that grid cell). A vehicle then maintains this belief (its trajectory distribution) and updates it at each timestep with its new state information and new future prediction. Note that the vehicles can use a partially-observable or fully-observable world model, as long as whichever model it uses is appropriately rolled into its predicted future route. Our experiments with robots use a partially-observable model, while our simulation experiments use a fully-observable model.

Problem Statement

- A *reservation* is a trajectory distribution that has been granted by the intersection manager; a *reservation request* is a message containing the proposed trajectory distribution.
- A *routing* R generated at time t is the set of all trajectory distributions that have been granted. $R_t = \{\rho(v_1, t), \rho(v_2, t), \dots\}$.

As we expect the vehicles to control their own low-level motion, we seek both a routing algorithm and a protocol. The algorithm will be executed centrally by the intersection manager to serve reservations to vehicles as they appear in an online fashion. The protocol will restrict the behavior of the vehicles and govern their communication with the intersection manager. Combined, the algorithm and protocol should result in the following properties: 1) vehicles will be *safe*, and not collide; 2) vehicles will be routed through the intersection *efficiently*, with little wait time; 3) vehicles will have near-autonomy, controlling their own motion as much as possible; and 4) communication will be limited to combat network saturation.

OUR APPROACH

While the intersection manager is centralized, we do not require that all vehicles motions are determined by the intersection manager. Instead, each vehicle controls its own motion, using whatever control policy it wishes. The only restriction on motion is that a vehicle cannot enter the intersection without a reservation. A vehicle obtains a reservation by requesting it from the intersection manager. It is the vehicle's responsibility to predict its motion, while it is the intersection manager's responsibility to accept or deny requests to ensure safety and fairness.

As a vehicle approaches the intersection, it negotiates a reservation with the intersection manager in several discrete stages. Below is a timeline that describes the stages of negotiation a vehicle undergoes and the necessary messages communicated between intersection manager and vehicle.

- 1 *Initial prediction*. When vehicle comes into the field-of-view, vehicle sends intersection manager its current trajectory distribution.
- 2 *Reservation request*. When vehicle reaches the safety zone threshold, it requests a reservation for its current trajectory distribution.
- 3 *Reservation request with waiting*. Until a reservation request has been granted, vehicle, while remaining in the safety zone, keeps requesting trajectory distributions (in whatever order it wishes, typically just by incrementing the time it enters the intersection).
- 4 *Proceed*. Once the reservation has been granted, the vehicle proceeds unhindered through the intersection.

A Safe Routing Algorithm

We will now discuss the question of safety, and then give several safe algorithms for solving the autonomous intersection management problem. We will begin with a simple, naive method that ensures safety under our model of uncertainty; we will then successively build on that method, at each step adding the desired features of low-latency routings, non-myopic routings, low-chatter communication, and computation efficiency.

Definition of safety A *collision* occurs when two vehicles occupy the same grid cell. We say a routing R is *safe with tolerance ϵ* if the probability of a collision is at most ϵ in every cell. Notationally, for all granted reservations $\rho(v), \rho(v') \in R$:

$$\rho(v) \cdot \rho(v') = 0 \quad \forall v \neq v' \quad (1)$$

We presume that allowing even the slightest possibility of a collision would preclude the system's adoption in the real world (the poor safety record of human drivers notwithstanding). As such, for the remainder of this paper, we require that the probability of a collision ϵ be exactly zero, and call such a routing *safe*.

Safety Zone We introduce a 'safety zone,' similar to the contingency plan method (Bekris & Kavraki 2007). The key idea is that a vehicle need not specify its future behavior exactly, as long as there exists a contingency plan allowing it to react safely to any circumstance. In our case, the contingency plan is for the vehicle to 'slam on the brakes,' or decelerate as quickly as possible. As long as the vehicle is outside the safety zone, this contingency plan can guarantee safety independent of all other vehicles in the world.

Specifically, the *safety zone* is the area in which the vehicle must have a reservation to guarantee safety. This region is the start of the intersection plus the *stopping distance*, the necessary distance to stop at maximum deceleration. Refer back to Figure 1(a) for a depiction of these two concepts.

We introduce a natural (though naive) extension of FCFS, which we call FCFS-hard, whereby the intersection manager

only grants a reservation request if the probability of it colliding with a prior reservation is zero. This process can be thought of as first ‘hardening’ the probability distribution for the predicted route ρ by treating any *possibility* of being in a grid cell as the *certainly* of being there, and then routing the hardened route $\tilde{\rho}$ via FCFS (as if it were deterministic / certain).

$$\tilde{\rho}(i, j, k) \equiv \lceil \rho(i, j, k) \rceil$$

FCFS-hard is extremely conservative in that it will cause vehicles to reserve the intersection for much longer than they need it and will lead to larger latencies.

Reducing Latency

To do better, we propose a ‘lazy’ reservation scheme, whereby vehicles make a reservation just-in-time (when they have the *least* uncertainty about their future motion). Then the hardened belief is reserved, as in FCFS-hard. To still ensure safety, we define *just-in-time* as the earliest timestep a vehicle has a nonzero probability of occupying a cell in the safety zone. Note we could pad the safety zone for additional caution, if, for example, we are unsure of our predictions or mechanical limitations; we would then trade off efficiency for additional safety. As we take the amount of padding to infinity, then we arrive at the pro-active FCFS algorithm again.

It is noteworthy that we’ve increased the model complexity to account for uncertainty. Our main assumption for safety, which has been implicit heretofore, is that vehicles can accurately predict a *probability distribution* over their future motion, as opposed to their *exact* future motion. This property is typically called *calibration*; the believed probabilities of motion match the actual probabilities, if measured empirically. While such well-informed models are unrealistic, we provide the following robustness guarantee for FCFS-hard: as long as the supports of all vehicles’ predicted routes contain the supports of their empirical route distribution, it is easy to see that FCFS-hard will still route them safely. In other words, as long as a vehicle admits some nonzero probability of occupying a cell, when the vehicle will in fact (empirically) occupy that cell with nonzero probability, then it will be routed safely — whether or not the probabilities match exactly.

Non-myopic Routing: the Optimization Problem

Heretofore, we’ve been implicitly trying to minimize the aggregate delay of vehicles through the intersection by using the greedy FCFS-hard method. We will now state explicitly the objective function and offer a non-myopic method for optimizing it.

Latency Intuitively, the *latency* of a vehicle, denoted l , is the time the vehicle must sacrifice at the intersection to coordinate with the other agents (via the intersection manager). If there were no other vehicles, this value would be zero. Formally, l is the difference between the expected exit time of the initially requested route, T^* , and the expected exit time of the granted reservation’s route, T (as predicted at t^* and t , respectively): $l(\rho(v, t)) = T - T^*$. Note that latency is always nonnegative, since we assume calibration of

predictions (i.e., the realized empirical distribution over trajectories is the same as the predicted distribution).

The *aggregate latency* L of a routing R is the sum of the latencies for each vehicle: $L_t(R_t) = \sum_v l(\rho(v, t))$. The intersection manager’s task is to find a routing that minimizes the aggregate latency subject to the constraints of safety. Although previous approaches (Dresner & Stone a) have examined throughput as well as latency, since we pose the problem as an optimization, it behooves us to define a single objective; naturally, low latency leads (generally) to high throughput, so this is our choice. Formally,

$$J_t(R_t) = \min_{R_t} L_t(R_t).$$

Reducing Computation

The algorithm must be able to plan online in real-time. As such, planning for the optimal solution (i.e., enumerating all possible routings to maximize J_t) is not an option. Instead, we can focus the search on just those vehicles that are near the intersection at t , which we call the *relevant* vehicles. Intuitively, we are factoring the problem into the subproblems of 1) routing the relevant vehicles at t , and 2) routing later vehicles. Hence, each optimization becomes agnostic to those at other times and maximizing over these sub-routings individually is much more reasonable, computationally.

The way we define *relevant* vehicles is important. At the one extreme, we could only include the entering vehicle as relevant; the result would be the FCFS policy exactly. On the other extreme, if we considered *all* vehicles as relevant, we’d be performing the maximization over the entire joint routing across all timesteps². We choose to define the vehicles *relevant to v* as those that might enter the intersection before v leaves it. Formally, $\text{RELEVANTVEHICLES}(v) = v \cup \{v' \mid v' \text{ has no reservation, } t_{\text{entry}}(v') \leq t_{\text{exit}}(v)\}$.

To find the best routing, we simply enumerate the different routing permutations of the relevant vehicles. A *routing permutation* $R_V(\sigma)$, over a set of vehicles V , is a routing created by reserving the trajectory distributions for each $v \in V$, one at a time, in order of the permutation $\sigma \in S_{|V|}$. A vehicle whose distribution conflicts with a previously granted reservation has its trajectory distribution translated to the earliest non-conflicting time. Note that the number of routings we must try is not enormous; it includes permutations of only the relevant vehicles, which is typically on the order of the number of lanes.

Reducing Communication

If we assume that communication between the intersection manager and vehicles is free, then it is clearly optimal to communicate at each timestep and thus maintain the most accurate predicted routes. However, if there are many vehicles sending updates at every timestep, then the wireless network would quickly become saturated. A saturated wireless network suffers from dropped packets and high network latency, leading to unsafe operation of the intersection or requiring a much more conservative safety zone to account for the added delays of network latency and retransmissions.

²for all visible vehicles, only

Algorithm 1 THE FHJAS INTERSECTION MANAGER ALGORITHM

{Reservation-serving Routine}	{Active Sensing Routine}
if receive reservation request ρ from v then $V' = \text{RELEVANTVEHICLES}(v)$ $P' = \text{UPDATEGAUSSIANBELIEF}(P, \rho^v)$ $R' = \text{ADDERESERVATION}(R, \rho^v)$ $[J, R^*] = \text{FINDBESTROUTING}(V', P, R)$ $R \leftarrow R'$ if v was first in R^* end if	if $\exists v, (x, y)$ in safety zone, then $V' = \text{RELEVANTVEHICLES}(v)$ $P' = \text{UPDATEGAUSSIANBELIEF}(P, \rho(v, t))$ $J = \text{FINDBESTROUTING}(V', P, R)$ $J' = \text{FINDBESTROUTING}(V', P', R)$ Perform query if $J' > J$ end if

Individual vehicles do not necessarily know what other vehicles are doing, so it naturally falls to the intersection manager to decide when to communicate or not (an active sensing task). For example, it might choose to communicate at each timestep on a lonely road with only one vehicle approaching, while it might do so much more rarely on a busy highway. To decide whether it is worth the cost of querying a vehicle, the intersection manager must compare the value of the routing given its current information and the expected value of the routing it would make after it makes the query; we also add a penalty term λ to the objective to model the cost of communication.³ In other words, it must solve for the *expected value* of the objective function under the posterior distribution R' : $J'(R') = \mathbb{E}_{R'}[L(R')] - \lambda$. The intersection manager will then perform the query only if $J'(R')$ is greater than the objective function given the intersection manager's current information $J(R_t)$.

However, for safety, we still require that vehicles request a reservation as they enter the safety zone. The intersection manager, then, performs both the tasks of: 1) serving reservation requests passively and 2) choosing to query vehicles via active sensing. The first operation is asynchronous (it doesn't know when a vehicle will arrive at the intersection and request a reservation), while the second is synchronous (each timestep, it evaluates whether or not to query vehicles for an updated trajectory distribution). The final algorithm, THE FHJAS INTERSECTION MANAGER ALGORITHM, is given listed as Algorithm 1.

RESULTS

Real Robots

We implemented the FCFS-hard algorithm with real robots, which was successful in preventing collisions. To our knowledge, this is the first demonstration of a working autonomous, multi-robot intersection in the real world. We used the robots from a robot soccer team in the small-size league of Robocup; see Figure 1(b). For maintaining beliefs, we also used the team's vision software, which relies on two overhead cameras, and the team's low-level motion control system. In essence, the software uses a bank of extended Kalman filters. We then utilized observed acceleration and velocity limits to obtain the necessary bounded belief support condition for our algorithms. Altogether, we used five

³Further modeling of communication latency explicitly is a complex task, and far outside the scope of this paper.

robots in a two-lane intersection.

Recall that the intersection management algorithm is agnostic to the behavior of the vehicles themselves; hence, we implemented a simple control policy, ASAP (As Soon As Possible), which we now describe informally (we omit insignificant details for clarity and brevity). Intuitively, this policy attempts to get the vehicle through the intersection as fast as possible, using maximum velocity and maximum acceleration, but coming to a stop at the intersection if it must wait for a reservation. We also included special logic to ensure vehicles don't run into vehicles ahead of them in the same lane (for example, several vehicles are 'backed up' at the intersection waiting for a reservation, so an approaching vehicle must stop behind those vehicles).

Despite our focus on the real world, we found that we did not have enough robots available to achieve congestion on a multi-lane intersection. For this reason, we found that the difference in latencies between the three safe algorithms was negligible (and that, indeed, all achieved collision-free operation). Hence, to scale up and compare the algorithms more thoroughly, we moved to a simulated domain.

Simulation

For the simulation, we simply implemented ballistic motion equations for velocity and position, used the same vehicle control policy as for the real robots, and proceeded by discrete timesteps. We imposed artificial constraints on acceleration and velocity. For each lane, we spawned new vehicles with a fixed probability, in order to simulate an infinite domain. To add randomness to the motion, we added a 'random walk' step at each timestep, either adding or subtracting a distance proportional to the vehicle's velocity. With this method, we maintained the desired characteristic of finite belief support.

We found that the original FCFS algorithm (that doesn't reason about uncertainty) is unsafe and resulted in collisions; every other algorithm (each of which uses the belief-hardening process described in Section) succeeded in preventing all collisions. Hence, we compare the performance of the other three algorithms, namely: 1) *FCFS-hard*, the naive algorithm that makes reservations as soon as possible; 2) *FHJ* (FCFS-hard Just-in-time), which makes reservations just-in-time and considers routings with relevant vehicles; and 3) *FHJAS* (FHJ with Active Sensing), which adds the active sensing routine that optimizes the communication-sensitive objective function.

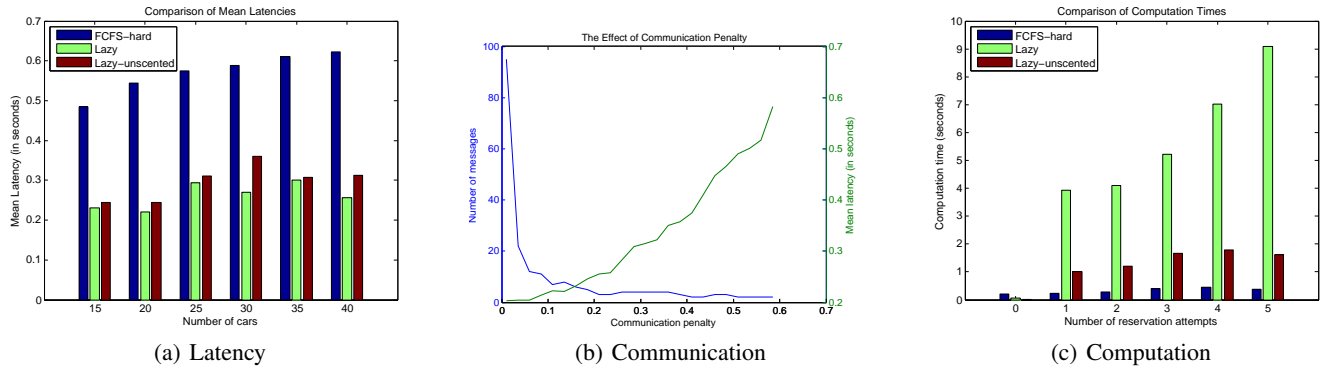


Figure 2: These figures compare the intersection manager algorithms according to latency, communication, and computation. 2(a) shows that the Lazy algorithm vastly outperforms the naive algorithm with respect to aggregate latency, while the Lazy-unscented algorithm incurs very little additional latency. 2(c) shows that the Lazy-unscented algorithm performs much more efficiently than the Lazy algorithm. 2(b) highlights the tradeoff between amount of communication and latency. In simulation, we have the ability to control the communication cost, but in reality it should be dictated by observed characteristics of the actual network. In essence, this figure validates the efficacy of our method to limit communication.

CONCLUSIONS AND FUTURE WORKS

In this paper, we’ve supplied several autonomous intersection control methods, each one satisfying an additional desideratum beyond the previous. We’ve built upon the simple strategy of FCFS to explicit reasoning about uncertainty, showing that doing so is necessary for safety. We’ve introduced a non-myopic optimization technique for traffic latency. We’ve added explicit reasoning about communication costs to combat wireless network congestion. We’ve demonstrated the efficacy of the autonomous intersection paradigm on real robots (a novelty, to our knowledge), as well as showing scalability of our methods on larger, simulated domains.

References

- Bazzan, A. 2005. A distributed approach for coordination of traffic signal agents. In *Proceedings of AAMAS-05*.
- Bekris, K., and Kavraki, L. 2007. A decentralized planner that guarantees the safety of communicating vehicles with complex dynamics that replan online. In *Proceedings of IROS-07*.
- Dresner, K., and Stone, P. Multiagent traffic management: A reservation-based intersection control mechanism. In *Proceedings of AAMAS-04*.
- Dresner, K., and Stone, P. Multiagent traffic management: An improved intersection control mechanism. In *Proceedings of AAMAS-04*.
- Jia, Z., and Ierapetritou, M. Short-term scheduling under uncertainty using MILP sensitivity analysis. *Ind. Eng. Chem. Res.*
- Jozefowska, J. 2007. *Just-in-time Scheduling*, volume 106 of *International Series in Operations Research and Management Science*.
- Muhuri, P., and Shukla, K. 2008. Real-time task scheduling with fuzzy uncertainty in processing times and deadlines. *Applied Soft Computing* 8(1).

Naumann, R., and Rasche, R. 1997. Intersection collision avoidance by means of decentralized security and communication management of autonomous vehicles. In *Proceedings of ISTATA-97*.

P. Hunt, D. Robertson, R. B., and Winton, R. 1981. SCOOT - a traffic responsive method of co-ordinating signals. Technical Report TRL-1014.

Rasche, R.; Naumann, R.; Tacke, J.; and Tahedl, C. 1997. Intersection collision avoidance by means of decentralized security and communication management of autonomous vehicles. In *Proceedings of ITSC-97*.

Roozmond, D. 1996. Intelligent traffic management and urban traffic control based on autonomous objects. In *Proceedings of AIS-96*.