In *The Third International Joint Conference on Autonomous Agents and Multiagent Systems* (AAMAS 04)
pp. 530-537, New York, New York, USA, July 2004.

Multiagent Traffic Management: A Reservation-Based Intersection Control Mechanism

Kurt Dresner* and Peter Stone
University of Texas at Austin
Department of Computer Sciences
Austin, TX 78712 USA
{kdresner, pstone}@cs.utexas.edu

Abstract

Traffic congestion is one of the leading causes of lost productivity and decreased standard of living in urban settings. Recent advances in artificial intelligence suggest vehicle navigation by autonomous agents will be possible in the near future. In this paper, we propose a reservation-based system for alleviating traffic congestion, specifically at intersections, and under the assumption that the cars are controlled by agents. First, we describe a custom simulator that we have created to measure the different delays associated with conducting traffic through an intersection. Second, we specify a precise metric for evaluating the quality of traffic control at an intersection. Using this simulator and this metric, we show that our reservation-based system can perform two to three hundred times better than traffic lights. As a result, it can smoothly handle much heavier traffic conditions. We show that our system very closely approximates an overpass, which is the optimal solution for the problem with which we are dealing.

1. Introduction

Traffic congestion is one of the leading causes of lost productivity and decreased standard of living in urban settings. Recent advances in artificial intelligence suggest that autonomous vehicle navigation will be possible in the near future. Individual cars can now be equipped with features of autonomy such as cruise control, GPS-based route planning [6, 8], and autonomous steering [5]. Once individual cars become autonomous, it is inevitable that before long all, or most, of the cars on the road will have such capabilities, thus opening up the possibility of considering autonomous interactions among multiple vehicles.

Multiagent Systems (MAS) is the subfield of AI that aims to provide both principles for construction of complex systems involving multiple agents and mechanisms for coordination of independent agents' behaviors [9]. In this paper, we propose an MAS-based approach to alleviating traffic congestion, specifically at intersections.

Current methods for enabling traffic to flow through intersections include building overpasses and installing traffic lights. However, the former is only worth the cost at the most congested intersections, and the latter can be quite inefficient, often requiring cars to remain stopped even when no cars are present on the intersecting road.

At this time, it is possible to create a system in which cars are driven by a central computer, and humans are merely passengers. Such a system would involve prohibitively expensive and inefficient communication and control infrastructure. Here we aim to maximize the efficiency of moving cars through intersections with minimal centralized infrastructure. We assume that intersections can be outfitted with a wireless communication system, and that they use our protocol for communicating with oncoming traffic and giving permission for cars to pass.

Cars must only traverse intersections when allowed to by the protocol (as they do today by obeying red and green lights), but otherwise are free to decide for themselves how to drive. That is, each car is treated as an autonomous agent, and in particular need not surrender control to any centralized decision maker.

Given the above assumptions, we propose a novel reservation-based system by which cars request and receive time slots from the intersection during which they may pass. In this paper, we present theoretical and empirical results relating the overpass, traffic signal, and reservation-based intersection control mechanisms. Each of these mechanisms is fully implemented in a custom traffic simulator, which we introduce here. Our results indicate that the reservation-based system allows traffic to flow through the intersection much more efficiently than the traffic light mechanism, and with efficiency approaching that of the overpass mechanism.

* The primary author of this paper is a graduate student.

2. The Model

Our model of intersection traffic is a somewhat simplified version of real-world intersection traffic. We do not allow cars to turn, and all cars begin traveling roughly the same speed. These simplifications make the analysis and implementation a lot easier, but do not detract from the fundamental challenges of the problem. Since we are trying to create a system in which vehicles reach their destinations in a minimum amount of time, it is not unreasonable to assume that each vehicle is always trying to travel at the speed limit. Additionally, almost every vehicle in production is capable of attaining the speed limit of any road, so this assumption is not a far stretch.

Once we make these assumptions, we next consider the question: How do we measure whether or not one intersection is better than another? Certainly, safety is a primary concern. An automated traffic control system will not be accepted unless the probability of collisions is extremely low. A secondary concern however is how efficient the intersection is. To measure intersection efficiency we consider specific metrics related to throughput and delay. Throughout this study we consider safety to be a prerequisite: all control policies we consider have 0 probability of collisions as long as the driving agents follow the protocols correctly.

2.1. Throughput

One metric we examine briefly is the amount of traffic that can be handled by an intersection. While this is hard to measure precisely, we make qualitative claims regarding the throughput of several different systems.

2.2. Delay

Delay is the primary metric we consider; what effect does the presence of the intersection have on the overall journey of a vehicle? Additionally, we would like to ensure that no vehicle's travel time is dramatically increased in order to save a few seconds for some other vehicle. We want to ensure that the average case is not bad, but we also want to ensure that the worst case is not *too* bad.

2.2.1. Average Delay Consider the set of all vehicles passing through an intersection in a period of time to be C. Assume that were there no other vehicles on the road, vehicle v_i would have made its trip from point A to point B in time $t_0(i)$. However, due to other cars in the same lane and due to having to go through an intersection involving other vehicles, v_i arrives instead at time $t(i)$. Then we define the *average delay* of an intersection to be:

$$\frac{1}{|C|}\sum_{v_i \in C} t(i) - t_0(i)$$

2.2.2. Maximum Delay The worst case delay is then:

$$\max_{v_i \in C} t(i) - t_0(i)$$

3. Overpass, Traffic Light Theory

Given these metrics, we can analyze the theoretical performance of some intersection control systems.

First, note that with an overpass the cars never have to slow down at the intersection. Thus, the average and maximum delay are both 0.

Analysis of the traffic light model is somewhat trickier. In this model, a vehicle has a random chance (at an isolated intersection) of making it through the intersection unimpeded. If it does not, the vehicle must come to a complete stop and wait, regardless of the amount of cross-traffic. The average and maximum delays are complicated functions of the timing of the lights (how long they are green and red), how many cars are on the road, and what the velocities of the other cars are, among other things. In order to analyze this model, we first make some simplifying assumptions.

1. *Cars traveling in the same direction do not interact with one another.*

   By allowing cars traveling in the same direction to pass through one another, we can remove car-car interactions from the model altogether. We are looking for a theoretical lower bound, and car-car interactions can only increase the delay on the average vehicle.

2. *Cars that have to decelerate due to a red light reach the intersection at full speed precisely when the light turns green again.*

   Again, as we are looking for a lower bound, we can give the driver agent more information than a normal traffic light would. This allows the driver agent to accelerate at just the right time such that it loses the minimum amount of time due to the traffic light.

Consider the following parameters:

P — the period of the traffic light

α — the fraction of the light's period that the light spends on green in one direction.

From the definitions of these parameters, we can specify two constraints:

$$P > 0 \quad \text{and} \quad 0 < \alpha \leq 1 \tag{1}$$

Because we have assumed the cars are independent, we can find the average delay per car by finding the expected delay for one car traveling alone. With the assumption that the vehicles always reach the green light at top speed, we remove any dependence on the acceleration or deceleration of the vehicle. To calculate the delay for one vehicle, we find the difference between the time the vehicle would have arrived at the traffic light were it always green and the time that the vehicle actually arrives at the traffic light.

In the end, we get an answer that is dependent only on α and P. The car reaches the green light without having to decelerate and experiences no delay at all α of the time. The remaining $(1-\alpha)$ of the time, the car does not reach the intersection until the beginning of the next cycle. Since the length of time for which the light is red is $(1-\alpha)P$, on average a car that does not make the green light will wait $\frac{1}{2}(1-\alpha)P$. Thus, the total expected delay for a car is:

$$\frac{1}{2}(1-\alpha)^2 P \qquad (2)$$

As α approaches 1, the expected delay approaches zero. As α approaches zero, the expected delay approaches $\frac{1}{2}P$. This is slightly counter-intuitive — common sense says the delay should grow without bound — but it is important to remember that at the exact moment the signal turns green (even if only for an instant), the vehicle is allowed to go. As the period of the light increases, our expected delay increases linearly — the same fraction of vehicles miss the light, and those vehicles have to wait longer.

The maximum delay can be calculated similarly. The longest amount of time for which a driver could wait is the length of time for which there is no green, or:

$$(1-\alpha)P \qquad (3)$$

4. The Simulator

To test our traffic control policies, we developed a time-based simulator to model each car individually. The simulator (not including the intersection manager) runs with two main parameters:

1. Number of lanes traveling in each of the four directions. For all experiments reported in this paper, the number of lanes for each direction was the same.
2. Probabilities of attempting to spawn a vehicle in each direction (independently) at the beginning of each time step. For almost all of our experiments, the probability for each direction was the same.

For our simulations, the simulator models an area that is 400 m × 400 m. Lanes are 3.5 m wide, and each vehicle is 2 m wide by 4 m long. Figure 1 shows a screenshot of the graphical display. Java applets showing the simulator in action can be viewed at `http://www.cs.utexas.edu/users/kdresner/papers/2004aamas/`. In each cycle of the simulator (which for our experiments represents one fiftieth of a second) the following events occur:

1. In each direction, vehicles are randomly spawned with a predefined probability. Once a vehicle is spawned, it is placed uniformly at random in one of the lanes traveling in that direction. If placing the vehicle in that lane and direction would cause the vehicle to be following another vehicle too closely (within 1 second[1] or 1 meter), the vehicle is not spawned. Our simulator spawns all vehicles traveling at the speed limit and never spawns a vehicle where it would be in danger of colliding with another vehicle.
2. The driver of each vehicle is given the distance to the car in front of it. This information could be sensed with on-board sensors such as cameras and/or range-finders and is used to avoid collisions with cars traveling in the same direction.
3. Each driver then takes an action based on this information: ACCELERATE (increase velocity by 3 m/s^2), DECELERATE (decrease velocity by 15 m/s^2), or COAST (maintain current velocity). The simulator enforces the invariant that $0 \leq \texttt{speed} \leq \texttt{top speed}$.
4. Each vehicle's position and velocity are updated according to the actions the driver took.
5. Any vehicles which have left the domain of the simulator are removed from the simulation. Each vehicle tracks its own delay, and this value is used to update the global value for total delay and maximum delay. The total number of vehicles which have finished their journey is also updated.

The behavior of the driver agent is completely independent of the simulator, and can be different for every vehicle, provided it adheres to our vehicle-intersection protocol. For our experiments, we used the same driver agent in each vehicle. It behaves as follows when it receives vehicle information from the simulator.

- COAST;
- If $\texttt{speed} < \texttt{speed limit}$, ACCELERATE;
- If less than 1 second or 1 meter behind the vehicle in front and $\texttt{speed} > 0$, DECELERATE;[2]
- If not through the intersection already, interact with the intersection as specified separately for each intersection control policy in Section 5;

As an intial test of the simulator, we begin by verifying that the empirical performance matches our theoretical predictions for the cases that can be analyzed completely, namely the overpass and traffic light intersection

1 A vehicle a is considered within t seconds of vehicle b if a is behind b and the distance between the back of b and the front of a is $velocity_a \cdot t$.

2 This is not done when vehicles are allowed to pass through one another.

In *The Third International Joint Conference on Autonomous Agents and Multiagent Systems* (AAMAS 04) pp. 530-537, New York, New York, USA, July 2004.

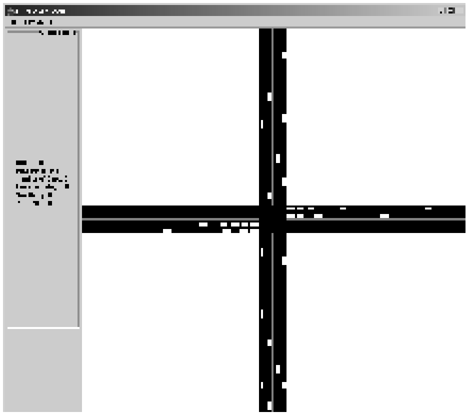

Figure 1. A screenshot of the graphical display of our simulator with 2 lanes in each direction. Cars traveling in the East/West directions are stopped at a red light.

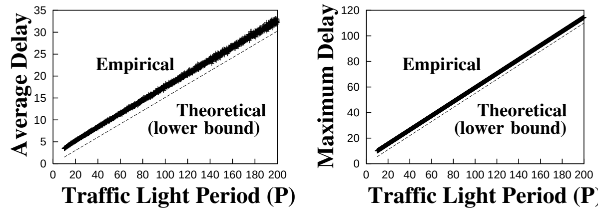

Figure 2. Average (left) and maximum (right) delay as a function of P for 1,000,000 steps of the simulation, spawning cars with probability .001.

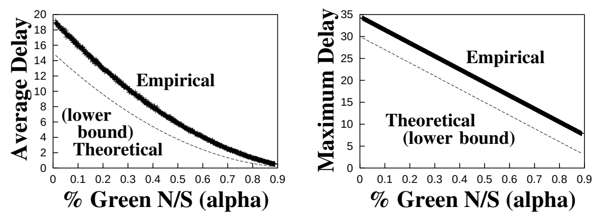

Figure 3. Average (left) and maximum (right) delay as a function of α for 1,000,000 steps of the simulation, spawning cars with probability .001.

controllers. We ran some simulations to determine both the average[3] and maximum delays as functions of both α and P. Figure 2 shows the average and maximum values which we obtained running the simulator for 1,000,000 steps with 1 lane in each direction, spawning cars with probability .002, letting α be .45, and varying P from 10 to 200 in steps of .1. Similarly, Figures 3 shows the average and maximum values obtained by fixing the period at 30, while allowing α to vary from .01 to .89 with steps of .001.[4] Because α is defined as the fraction of the time the light is green in the North/South directions, we only run the simulator with cars spawning in the North/South directions. The graphs for the East/West directions are just a horizontal reflections of these data.

Notice that the empirical data is consistent with our theoretical predictions modulo the fact that our driver agent does not have the ability to reach every green light at full speed. The discrepancies between the lines are consistent with our expectations — they can be attributed to the time it takes for a vehicle to come to a stop at the light and then accelerate again to full speed. Thus we conclude that the simulator and driving agents operate correctly.

5. Intersection Control Policies

Using our simulator, we evaluated the performance of three different intersection control policies: the overpass, the traffic light, and the reservation system.

5.1. Overpass

The overpass is by far the simplest of the three policies: it lets vehicles through all the time. While there is no explicit third dimension in the simulator, by allowing vehicles traveling in orthogonal directions to pass through one another, the same results can be achieved. If the ability to turn is not required (as in our current study) the overpass is an optimal solution. The only delays are caused by cars traveling the same direction, which would happen even if there were not an intersection through which to travel.

5.2. Traffic Light

The traffic light model is a close approximation of actual traffic signals used on real roads. The model has three parameters: the period of the light system, the time between green lights in which all four directions' lights are red (expressed as a fraction of the period), and the time for which the North/South lights are green (again as a fraction of the period). North and South lights are always identical, as are East and West lights. Yellow lights are not necessary in our system. In real-world traffic signals, yellow lights alert the drivers as to when the next red light is coming. However in

3 Unless otherwise stated, all averages are over all cars spawned in one run of the simulation.

4 Java applets of showing our experiments can be viewed at `http://www.cs.utexas.edu/users/kdresner/2004aamas/`

our model, the drivers can actually query the light to determine in which state the light will be at a given time. This improves the ability of the traffic light to direct the vehicles through the intersection. It is important to note that simplifying assumption 2 from Section 3 does not apply to this model — it is specific to the driving agent and not a property of the intersection control mechanism. Our driver agent does not time the green lights perfectly, although with the traffic light system we implemented, such a driver agent could be created.

The interaction between the driver agent and the traffic light is straightforward. First, the driver calculates when it will reach the traffic light given its current velocity. The driver then sends a message to the intersection informing it of the time at which the driver expects to arrive. The intersection then responds with the range of times during or after the time specified by the driver, at which the lights will be green. The driver can then make any adjustments necessary to ensure that the vehicle enters the intersection when the lights are green.

5.3. Reservation System

In creating the reservation system we defined both the behavior of the system as well as the behavior of the driver agent when interacting with the system.

5.3.1. Behavior of the System The reservation system, which we introduce here, allows the driver agents to "call ahead" and reserve the spaces they will need. The intersection is divided into an $n \times n$ grid of reservation tiles, where n is called the *granularity* of the reservation system. Each tile can be reserved by one car per time step.

To use the reservation system, the car sends a message containing several parameters.

1. The time the vehicle will arrive
2. The velocity at which the vehicle will arrive
3. The direction the vehicle will be facing when it arrives
4. The vehicle's maximum velocity
5. The vehicle's maximum and minimum acceleration
6. The vehicle's length and width

From these parameters, the intersection simulates the journey of the vehicle through the intersection with the parameters provided, noting which cells will be occupied by the vehicle at each time step (as well as a few time steps before and after, for safety). If any of these cells is already reserved, the intersection rejects the driver's request. Otherwise, it accepts the driver's request.

5.3.2. Behavior of the Driver Agent At the beginning of every cycle, the driver agent determines when it expects to reach the intersection, where it will be when it first enters the intersection, and how fast it will be going.

If the driver has not yet made a reservation, it sends the intersection a message (as described above). If the intersection accepts the request, the driver agent notes that a reservation has been made (along with the parameters). If the intersection rejects the request, the driver decelerates and tries again at the next time step.

If the driver has made a reservation, it determines whether or not it can keep the reservation. reservation. If it determines that it can not meet the reservation, it cancels the reservation and the reservation-making process begins again.

6. Empirical Results

In this section, we evaluate the performance of our reservation system against both the overpass and traffic signal systems for varying amounts of traffic, numbers of lanes, and granularities of the reservation system.

First, it is important to note the amount of traffic that each system can handle. The overpass system can handle as much traffic as we can generate with our simulator, since it does not cause the cars to slow down at all. It is interesting, though to examine the throughput of the traffic signal system. One problem we had was that if we create traffic faster than we can move it through the intersection, it builds up until the simulator cannot simulate it all. For this reason, the simulator stops adding vehicles when it gets full.

Qualitatively, the reservation-based system is able to sustain a much higher throughput than any of the traffic signal systems before causing the system to become full.

Secondly, and more precisely, we compare the average and maximum delays when using the different intersection control policies.

6.1. Overpass: The Lower Bound

In our simulator, the overpass is as ideal as possible — it never requires any vehicle to slow down at the intersection. Furthermore, since all vehicles travel at the speed limit, vehicles traveling the same direction are never required to slow down for each other. For this reason, every vehicle using the overpass system experiences 0 delay. We ran many trials with the overpass system again to verify that our simulator worked as expected. By varying the top speeds of vehicles, we can produce delays with the overpass system, but in real-world situations, vehicles' top speeds are much higher than the speed limit.

6.2. Traffic Light

To evaluate the delay of the traffic light model, we set several different periods and ran a million steps of the simulator for an increasing car spawning probability — from .001 to .02. The graph in Figure 4 shows that for lighter traffic, a shorter period is strictly better. This is already known and is used daily in many cities, where late at night the period of traffic lights is reduced. As the traffic increases, higher periods are more efficient. Eventually, however, they cannot handle the traffic, and the delay begins to increase very rapidly. In Figure 4, the delay levels off because at this point the cars are so backed up that the simulation can not create any more of them.

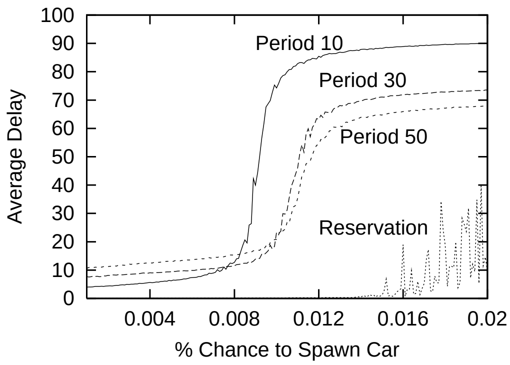

Figure 4. Average delays for traffic light systems with period 10, 30, and 50 seconds plotted against varying traffic levels along with a 1-tiled reservation-based system. Each direction has 1 lane.

An interesting thing to note is that even with only one tile, the reservation-based system does not break down until reaching a much higher level of traffic. An overloaded reservation system is chaotic: exactly when and how it breaks down varies wildly with when cars are spawned and which directions they are traveling. For that reason, running the simulation for a fixed number of steps can produce very different amounts of total delay. This explains why the line in Figure 4 representing the single-tile reservation system becomes so jagged.

6.3. Reservation System

To demonstrate the drastic improvement in throughput with the reservation system, we ran the same traffic amounts as in the traffic signal case, but this time used reservation systems with granularity one and two. In Figure 5, we show just how much of an improvement we can obtain over the single-tile case by increasing the granularity to just two. The amount of traffic handled by this system is much higher, and the associated delays are much lower. Note that with a granularity of two, it is permissible to have more than one car in the intersection at the same time: just not in the same quadrant.

After determining that the reservation system could manage as much traffic as a traffic light, we experimented along two main dimensions. First, we investigated the ability of

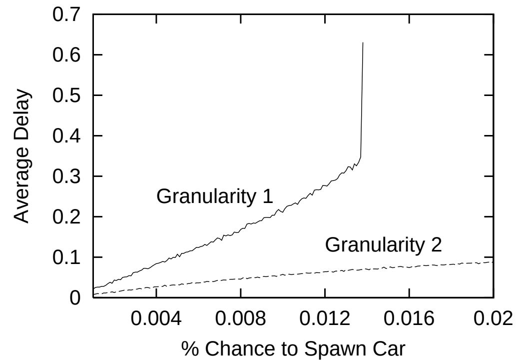

Figure 5. The average delay for 1- and 2-tile reservation systems with 1 lane per direction and varying traffic levels. Each data point represents 1,000,000 steps of simulation.

our system to scale up to wider roads (i.e. multiple lanes going in each direction). Second, we studied the impact of the main parameter in our approach, namely the granularity used on the part of the intersection when assigning reservations. In particular, we characterize the tradeoff in terms of computational complexity vs. performance level.

6.3.1. More lanes One question that arises is whether or not this system scales up to a larger number of lanes. We tested it up through a 6×6 intersection (6 lanes traveling in each direction), and it performed well in all the cases, averaging less than one half of one percent the delay with the traffic light system. We present these results in Figure 6. Recall that the overpass system, representing the lower bound, leads to 0 delay.

	Reservation			Light	
Lanes	Gran.	Avg	Max	Avg	Max
1	1	0.016	0.912	5.847	15.526
2	2	0.017	0.925	5.488	15.536
3	3	0.023	1.435	5.482	15.506
4	4	.019	1.590	5.351	15.536
5	5	0.031	1.902	5.439	15.506
6	6	0.025	1.926	5.378	15.517

Figure 6. Statistics for different numbers of lanes and granularities of the reservation-based system along with numbers from a traffic light system with period 20 seconds. The simulation was run for 1,000,000 steps with a car-spawning probability of .001.

6.3.2. Reservation Granularity The only parameter currently accepted by the reservation system is the side length of the grid representing the intersection, the granularity. A

In *The Third International Joint Conference on Autonomous Agents and Multiagent Systems* (AAMAS 04)
pp. 530-537, New York, New York, USA, July 2004.

granularity of 1 means a 1×1 grid of reservation tiles. A granularity of 2 means a 2×2 grid, and so forth. This leads to another tradeoff - computation complexity and fault-tolerance versus efficiency. With fewer reservation tiles, the intersection is more exclusive. With only 1 tile, for example, only one car may be in the intersection at a time, however with an extremely large number of tiles, the intersection reserves only enough space for the car, leaving little room for mistakes. As we will show, having only one tile can also lead to deadlock problems, whereas having a large number of tiles gives the intersection more flexibility with which to guide the cars. As for computation complexity, both the amount of calculation needed to process a single reservation and the space needed to hold the reservation grid increase as the square of the granularity.

	Delay		
Gran.	**Avg.**	**Max**	**Std Dev.**
2	0.109	1.698	0.239
3	0.131	2.750	0.274
4	0.043	1.026	0.134
5	0.071	1.812	0.185

Figure 7. Simulation statistics for a reservation system with varying granularity. There are 2 lanes in each direction and the car-spawning probability is .001.

One problem we discovered occurs whenever cars traveling in the inside lanes, but in opposite directions are competing for the same tiles. This happens usually when the granularity is an odd number. If one car has to slow down because it can not obtain a reservation, when it finally does get a reservation it will occupy those tiles for a longer period of time. This makes it more likely that the next car coming in the opposite direction will have to slow down even more. This process eventually slows the cars down more and more. For small to average amounts of traffic, this causes larger delays. For very heavy traffic, it will eventually deadlock the intersection. As shown in Figure 7, with 2 lanes going in each direction it is better to have a 2×2 grid rather than a 3×3 grid. Increasing to a 4×4 grid is better than 2×2, but increasing it to 5×5 is again worse. The deadlocking effect is hard to measure quantitatively, because if the cars start to deadlock, they begin to make reservations for very long periods of time — so long, in fact, that they overflow the memory of the computer system.

Because of this last problem, the reservation system should always be run with a granularity at least as high as the number of lanes, and ideally a multiple of the number of lanes. To test the effect of higher granularities, we ran the simulator with varying numbers of lanes and granularities. As Figure 8 demonstrates, more lanes require a higher granularity (though even with low granularity, this system out-performs the traffic light system). However, once the granularity is greater than the number of lanes traveling in each direction, subsequent increases in granularity provide only a marginal benefit, if any. Figure 9 shows a typical moment during a simulation with 6 lanes going in each direction and a granularity of 48.

	Granularity				
Lanes	**1**	**2**	**6**	**12**	**48**
1	0.016	0.005	0.007	0.005	0.007
2	0.059	0.017	0.010	0.008	0.003
3	0.142	0.038	0.009	0.011	0.013
6		0.132	0.025	0.011	0.006

Figure 8. Average delays for the reservation system with independently varying numbers of lanes and granularity. All simulations run for at least 500,000 steps. 6 lanes with 1 tile breaks down and overflows the system memory before 500,000 steps can complete.

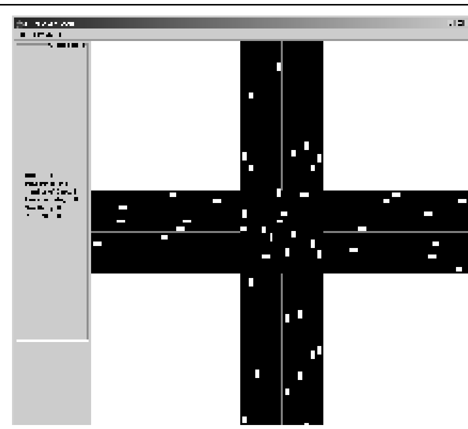

Figure 9. A screenshot of with 6 lanes in each direction and a granularity of 48. Note that there are cars traveling all four directions in the intersection simultaneously.

7. Discussion and Related Work

As shown in the previous section, the reservation-based approach drastically outperforms the traffic light system. However, there would be many challenges associated with creating such a system for the real world. Additionally, within the simulator, there are assumptions in the work reported here that can be relaxed in future work.

7.1. Transfer to the Real World.

Our simulator functions partially on the idea that the driver agent for each vehicle is using whichever intersection management policy is in place. This isn't much different from the real world — if someone wants to run a red light, there is nothing in the vehicle to stop them from doing so. However, it is in everyone's best interest for each driver to obey the traffic signals that he or she sees. A challenge that remains, however, is modifying the system such that a human driver could use it. Currently, the margins for error in the system are too small for a human to be able to control the vehicle. The margins of error could be enlarged, but the efficiency benefits associated with the system stem directly from the ability of a computer driver agent to precisely control the vehicle. Were there a few human drivers mixed in with the rest, a signal could be sent ahead to let the intersection know a human driver was approaching. The intersection could then make some extra room for the driver to maneuver through the intersection. However, with a large percentage of human drivers, a traffic light would probably be more appropriate.

7.2. Related Work

Most study of intersection management has been from a theoretical perspective, reducing the problem to scheduling jobs that may be competing for mutually exclusive resources. Irani and Leung create what they call "conflict graphs" in which nodes competing for the same resources share an edge. They then search these graphs for independent sets [1]. Later, they present a probabilistic approach to the problem, and report actual simulated results using this approach [2]. Their approach, however does not aim to improve the underlying system of traffic signals, but rather to tune them more effectively.

As far as multiagent approaches, most prior work has focused on increasing the throughput and decreasing delays for traditional traffic light systems. For example, Roozemond allows intersections to act autonomously, sharing the data they gather [7]. The intersections then use this information to make both short- and long-term predictions about the traffic and adjust accordingly. This approach still assumes human-controlled vehicles — the agents in this case are only the intersection controllers themselves.

Coming from the other direction, agent-based controllers have been designed for mechatronic systems, which are defined as "complex technical systems whose motion behaviour is actively controlled with the help of computer technology" [4]. They point to intersection management as one application of such controllers.

Finally, Kolodko and Vlacic describe a primitive system for intersection control which is very similar to the reservation system with granularity 1 [3]. This system has been successfully implemented on real autonomous vehicles.

8. Conclusion

This paper makes two main contributions. First, it describes and specifies a custom intersection simulator for modeling intersection control policies along with precise metrics for measuring the efficiency of these policies. Second, it proposes a reservation-based multiagent control policy that dramatically out-performs a traffic light policy and approaches the theoretical optimum in simulation. This policy is fully implemented and tested in simulation.

Current limitations include the inability of vehicles to turn and the constraint that vehicles may not change their velocity while in the intersection. Relaxing these limitations is a part of our on-going research agenda. Once autonomous vehicles are common, this mechanism may be useful for managing real traffic.

Acknowledgments

This research was supported in part by NSF CAREER award IIS-0237699.

References

[1] Irani and Leung. Scheduling with conflicts, and applications to traffic signal control. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1996.

[2] Irani and Leung. Probabilistic analysis for scheduling with conflicts. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1997.

[3] J. Kolodko and L. Vlacic. Cooperative autonomous driving at the intelligent control systems laboratory. *IEEE Intelligent Systems*, 18(4):8–11, July/August 2003.

[4] O. Oberschelp, T. Hestermeyer, B. Kleinjohann, and L. Kleinjohann. Design of self-optimizing agent-based controllers. In *Proceedings of the 3rd International Workshop on Agent-Based Simulation*, 2002.

[5] D. A. Pormerleau. *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers, 1993.

[6] S. Rogers, C.-N. Flechter, and P. Langley. An adaptive interactive agent for route advice. In O. Etzioni, J. P. Müller, and J. M. Bradshaw, editors, *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 198–205, Seattle, WA, USA, 1999. ACM Press.

[7] D. A. Roozemond. Using intelligent agents for urban traffic control control systems. In *Proceedings of the International Conference on Artificial Intelligence in Transportation Systems and Science*, pages 69–79, 1999.

[8] T. Schonberg, M. Ojala, J. Suomela, A. Torpo, and A. Halme. Positioning an autonomous off-road vehicle by using fused DGPS and inertial navigation. In *2nd IFAC Conference on Intelligent Autonomous Vehicles*, pages 226–231, 1995.

[9] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, July 2000.