Learning Cooperative Lane Selection Strategies for Highways

David E. Moriarty
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina Del Rey, CA 90292
moriarty@isi.edu

Pat Langley
Daimler-Benz Research and Technology Center
1510 Page Mill Road
Palo Alto, CA 94304
langley@rtna.daimlerbenz.com

Abstract

This paper presents a novel approach to traffic management by coordinating driver behaviors. Current traffic management systems do not consider lane organization of the cars and only affect traffic flows by controlling traffic signals or ramp meters. However, drivers can increase traffic throughput and more consistently maintain desired speeds by selecting lanes intelligently. We pose the problem of intelligent lane selection as a challenging and potentially rewarding problem for artificial intelligence, and we propose a methodology that uses supervised and reinforcement learning to form distributed control strategies. Initial results are promising and demonstrate that intelligent lane selection can achieve higher traffic throughput, maximize desired speeds, and reduce the total number of lane changes.

Introduction

A large effort is under way by government and industry in America, Europe, and Japan to develop intelligent vehicle and highway systems (IVHS). These systems incorporate ideas from artificial intelligence, intelligent control, and decision theory, among others, to automate many aspects of driving and traffic control. The goals of IVHS are quite broad and include increased traffic throughput, fewer accidents, reduced fuel consumption, and a better driving experience.

The work in this paper targets one component of the overall task: the problem of managing traffic. Advanced traffic management systems are designed to reduce congestion and increase overall traffic throughput. Almost all such systems maintain efficient traffic flows by controlling traffic signals or highway ramp meters, treating traffic as a single mass and normally ignoring the behavior of individual cars (Gilmore, Elibiary, & Forbes, 1994; Kagolanu, Fink, Smartt, Powell, & Larson, 1995; Pooran, Tarnoff, & Kalaputapu, 1996). This view, however, misses an important component of traffic management: coordination of the cars themselves. Surprisingly, very little research has addressed how the cars themselves can sense and intelligently affect traffic flows. Drivers generate local behaviors such as lane changes and speed control. These behaviors could be coordinated to better maintain desired speeds and achieve greater traffic throughput.

Our view is that cars are not blind tokens, but rather can sense their environment and act intelligently and cooperatively to achieve a desired global behavior. More specifically, cars can learn to organize themselves by traffic lanes to increases overall traffic throughput, reduce the average number of lane changes, and maintain the desired speeds of the drivers. Intelligent lane selection should therefore complement existing efforts in advanced traffic management by providing better throughput in between traffic signals and better-defined driver behaviors for traffic-flow prediction.

A challenging problem for artificial intelligence and machine learning lies in the development of cooperative driving strategies for traffic management. This paper explores one form of this problem: intelligent lane selection. Each car receives local input of the surrounding traffic patterns and the desired speed of the driver, then outputs the lane in which to drive. A car's lane selections should consider not only the maintenance of its own desired speed, but also how the selection will affect the speeds of other cars. In this way, the cars should organize themselves into a cooperative system that lets the fast drivers pass through, while still letting the slow drivers maintain their speeds.

The work in this paper is exploratory in nature, and follows Dietterich's (1990) model for exploratory machine learning research. We formulate the problem of traffic management from a car-centered, machine learning perspective and present initial results of cooperative lane selection.

Problem Definition

We recast the traffic management problem as a problem in distributed artificial intelligence, where each car represents an individual agent in a multi-agent system. Cars act on their world (highway) by selecting appropriate lanes to drive in. They interact with other cars by competing for resources (spaces on the highway). Each action is local in nature, and may not produce any noticeable benefit to the car. Collectively, however, the local actions can improve the global perfor-

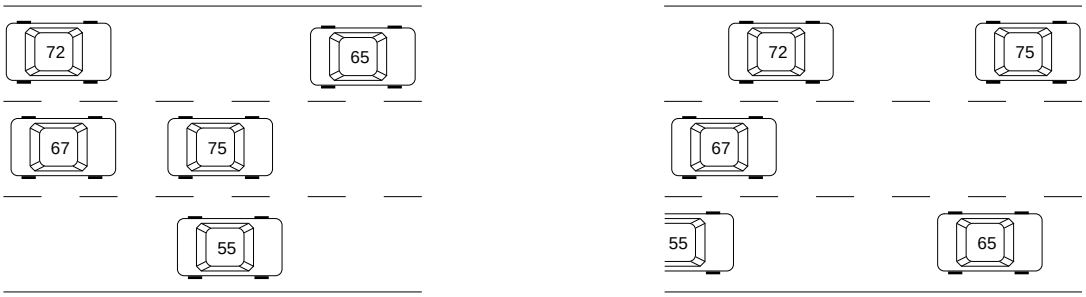


Figure 1: (a) An example traffic situation in which the traffic flows from left to right and the number on each car shows the car's speed. (b) Traffic after reorganization in which car 75 and 65 swap lanes.

mance of the traffic. For example, yielding a lane to a faster car does not produce any local benefit to the slower car, but does increase the overall traffic throughput and let the passing car maintain its desired speed.

Figure 1(a) illustrates a situation where lane coordination is beneficial. The figure illustrates five cars along with their speeds, which will be used as identifiers. Car 72 is approaching car 65 and is unable to pass because of the position of car 67. Without reorganization, car 65 forces car 72 to reduce its speed and wait for car 67 to pass car 65, which will decrease traffic throughput and car 72's satisfaction. A solution is for car 75 and car 65 to swap lanes, followed by car 65 moving into the bottom lane. This maneuver ensures that no speeds are reduced and no throughput is lost.

Global traffic performance could be defined in many different ways. Governments want high traffic throughput, whereas drivers want to maintain desired speeds with few lane changes. We selected the driver-oriented metric, since drivers are likely to be the harshest critics of cooperative driving. The performance function P we devised for a set of cars C is given by the equation:

$$P(C) = \frac{\sum_{t=1}^{T}\sum_{i=1}^{N}(S_{it}^{a} - S_{it}^{d})^2}{TN} - \frac{60\sum_{i=1}^{N} L_i}{TN}, \quad (1)$$

where T is the total time steps (in seconds), N is the number of cars, S_{it}^{d} is the desired speed of car i at time t, S_{it}^{a} is the actual speed of car i at time t, and L_i is the total number of lane changes for car i over T time steps. The goal is to minimize the difference between actual speeds and desired speeds averaged over several time steps and over all cars. Each speed difference is squared to penalize extreme behavior. For example, driving 60 m/h 90% of the time and 10 m/h 10% of the time gives an average of 55 m/h but is clearly less desirable than driving 56 m/h 50% and 54 m/h 50% of the time, which also gives an average of 55 m/h. To discourage excessive lane changes, the performance function is adjusted by subtracting the number of lane changes per minute averaged over all cars.

The problem is thus to find a strategy or a set of strategies to maximize equation 1. A naive strategy for each car, which most traffic management systems assume, is to select the lane that lets it most consistently achieve its desired speed and only change lanes if a slower car is encountered. The disadvantage of such a strategy is that it does not take into account the global criteria of traffic performance. A slow car should not drive in the "fast" lane simply because it can maintain its desired speed. We will refer to cars that employ the naive strategy as *selfish cars*, since they maximize the local performance of their respective car. We are interested in strategies that maximize the aggregate performance of traffic. Cars that employ cooperative selection strategies will be termed *smart cars*.

Ideally, the smart cars should coexist with current drivers on the highways. This situation poses interesting research questions. How many smart drivers are necessary to make cooperation worthwhile? How quickly does the system break down when selfish drivers are introduced in the system? The experimental evaluation section presents some evidence that, even in distributions as high as 95% selfish cars, cooperative lane selection can improve traffic performance.

Communication and Coordination

The previous section defined the problem of car-centered traffic management, but left open some important issues in designing a distributed artificial intelligence system for traffic management. Specifically, we left open the level of communication between the cars and the amount of knowledge available on other cars' state. The multi-agent literature is often divided on these matters, and we feel that it is not central to the problem definition. Here we describe our assumptions about communication and state information.

We assume that cars have access to information on their own state, including knowledge of their current driving speed and the driver's desired speed. One could imagine a driver specifying desired speeds at the start of a trip, or the system could infer this information from the driver's historical behavior. We also assume that cars can perceive limited state information of surrounding cars, such as their relative speeds. The sys-

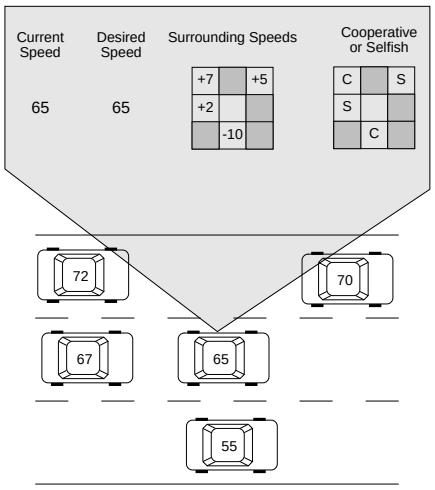


Figure 2: An illustration of the input to each car.

tem could sense this information using radar or receive it directly from other cars via radio waves or the Internet. Cars should also sense which surrounding cars are cooperative and which are selfish.

Figure 2 illustrates the input for a car in a specific traffic situation. The middle car receives as input its current speed, its desired speed, the relative speeds of surrounding traffic, and whether surrounding cars are cooperative or selfish. The range and granularity of the relative speed inputs could be adjusted to take into account both local traffic and upcoming traffic.

Note that the cars only receive a partial view of the overall traffic situation. Another design option is to give all of the cars complete information of all other cars and treat the problem as a global optimization problem. We selected the local input representation for two reasons. First, we believe that it is more realistic to assume that only local traffic infomation is available. Second, we believe that a local policy will provide more effective generalization across different highways.

We assume that the controller's output consists of three options: (1) stay in the current lane, (2) change lanes to the left, or (3) change lanes to the right. The output does not specify the best lane to drive in, but rather whether the lanes immediately left or immediately right are better than the current lane. This control provides flexibility, since it does not depend on the number of lanes on the roadway or knowledge of the current driving lane.

We assume that the controller's output represents a ranking of the three possible choices, with the highest ranked choice that is both valid and safe selected as the car's next action. For a choice to be valid, there must be a lane available in the specified direction. For a choice to be safe, there must not be a car in the same longitudinal position in the new lane. We assume that it is always safe to remain in the current lane.

Another important issue concerns the representation of the different types of lane-selection strategies. Clearly, different types of drivers should select lanes differently. Slower drivers will normally change lanes to create openings for faster traffic. Faster drivers change lanes to pass through slower traffic. Maintaining explicit control policies for each type of driver, however, requires *a priori* knowledge of the number of driver types and the boundaries that separate them. We chose instead to maintain a single control policy which takes as input the type of driver (desired speed input). Each car thus contains the same control policy, but since it receives driving style as input, it behaves differently for different drivers.

Learning Distributed Control Strategies

Creating distributed lane-changing controllers by hand appears quite difficult. It is unclear whether experts exist in this domain and, even if they do, experts often find it difficult to verbalize complex control skills, which creates a *knowledge acquisition bottleneck*. Also, the innumerable traffic patterns and varying driving styles create a very large problem space. Even with significant expert domain knowledge, hand crafting a controller that operates effectively in all areas of the problem space may not be feasible.

Our solution is to apply machine learning to develop controllers through experience with the domain. Unfortunately, the lane-selection problem appears out of reach of the more standard, *supervised* machine learning methods. Supervised learning would require examples of correct lane decisions, which are difficult to obtain without expert domain knowledge. We also do not want to mimic real drivers, since we believe that drivers do not currently select lanes cooperatively.

We chose a multi-level learning approach that capitalizes on the available domain knowledge, but is also flexible enough to learn under sparse reinforcements. The learning system consists of three main components: reinforcement learning using SANE, supervised learning from pre-existing domain knowledge, and a local learning strategy that is similar in spirit to temporal difference methods. Figure 3 illustrates the interaction of the different learning methods, which are described in the next three sections.

Reinforcement Learning using SANE

The backbone of the learning system is the SANE reinforcement learning method (Moriarty & Miikkulainen, 1996; Moriarty, 1997). This section gives a brief outline of SANE and its advantages; the aforementioned references provide more detailed information.

SANE (Symbiotic, Adaptive Neuro-Evolution) was designed as an efficient method for forming decision strategies in domains where it is not possible to generate training data for normal supervised learning. SANE maintains a population of possible strategies, evaluates the goodness of each from its performance

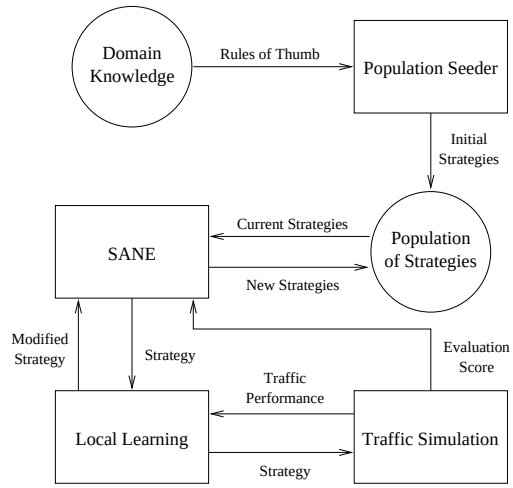


Figure 3: The organization and interaction of the different learning modules.

in the domain, and uses an evolutionary algorithm to generate new strategies. The evolutionary algorithm modifies the strategies through genetic operators like selection, crossover, and mutation (Goldberg, 1989).

SANE represents its decision strategies as artificial neural networks that form a direct mapping from sensors to decisions and provide effective generalization over the state space. The evolutionary algorithm searches the space of hidden neuron definitions, where each hidden neuron defines a set of weighted connections between a fixed input and fixed output layer. In other words, SANE evolves all of the connections and weights between the hidden layer and the input and output layers in a three-layer network.

SANE offers two important advantages for reinforcement learning that are normally not present in other implementations of neuro-evolution. First, it decomposes the search for complete solutions into a search for partial solutions. Instead of searching for complete neural networks all at once, solutions to smaller problems (good neurons) are evolved, which can be combined to form effective full solutions (neural networks). In other words, SANE effectively performs a problem reduction search on the space of neural networks.

Second, the system maintains diverse populations. Unlike the canonical evolutionary algorithm that converges the population on a single solution, SANE forms solutions in an *unconverged* population. Because several different types of neurons are necessary to build an effective neural network, there is inherent evolutionary pressure to develop neurons that perform different functions and thus maintain several different types of individuals within the population. Diversity lets recombination operators such as crossover continue to generate new neural structures even in prolonged evolution. This feature helps ensure that the solution space will be explored efficiently throughout the learning process.

SANE represents each lane-selection strategy as a neural network that maps a car's sensory input into a specific lane-selection decision. Each network consists of 18 input units, 12 hidden units, and 3 output units. A network receives input on the car's current and desired speeds and the speeds of surrounding traffic, and it outputs a ranking of the three possible choices.

A strategy is evaluated by placing it in a traffic simulator and allowing it to make lane changes in a certain percentage of the cars. Each strategy is evaluated independently of other strategies in the population. The fitness of a strategy is measured using equation 1 after some number of simulated seconds. SANE uses these evaluations to bias its genetic selection and recombination operations towards the more profitable lane-selection strategies.

Incorporating Domain Knowledge

The second learning component capitalizes on pre-existing domain knowledge and gives SANE a good starting set of initial strategies. Although expert information is difficult to obtain in this problem, general rules of thumb are not. For example, one good heuristic specifies that a very slow driver should in general not drive in the far left lane. Supervised learning from these general rules will not generate optimal lane selection strategies, but it can give the learning system a good head start towards intelligent behavior.

The population seeder applies such heuristics in the traffic simulator and generates a series of input and output pairs, which represent decisions made from the rules of thumb based on specific sensory input. These pairs denote examples of good behavior that can be fed to a supervised learning method to form initial strategies. Since SANE's strategies are represented as neural networks, the population seeder employs the backpropagation algorithm (Rumelhart, Hinton, & Williams, 1986) to train the networks over the training examples. To maintain diversity within the initial population of neural networks and not overly bias SANE toward the rules of thumb, only a subset of the networks are seeded using the default knowledge. In practice, we seed 25% of the initial population.

We used four rules to seed SANE's population of strategies:

- If your desired speed is 55 m/h or less and the right lane is open, then change lanes right
- If you are in the left lane, a car behind you has a higher speed, and the right lane is open, then change lanes right
- If a car in front of you has a slower current speed than your desired speed and the left lane is open, then change lanes left.
- In the previous situation, if the left lane was not open but the right lane is, then change lanes right.

These rules are based on our interpretation of the "slower traffic yield to the right" signs posted on the highways. We will refer to this strategy hereafter as the *polite strategy*. The selfish strategy described earlier in the paper operates using only the last two rules.

Local Learning

We also implemented a local learning module that, like the population seeder, was included to increase learning efficiency and thereby reduce the amount of simulation time necessary to form good strategies. Local learning occurs during the evaluation of a lane-selection strategy and makes small refinements to the strategy based on immediate rewards or penalties. A reward or positive training signal is given if there is a significant increase in traffic performance and a penalty or negative training signal is given if there is a significant decrease. In practice, traffic performance is sampled every 10 simulated seconds and a reward or penalty is generated if the difference in performance from equation 1 is larger than 10.

If a training signal is generated, all actions performed in the sampling interval are considered responsible. If the signal is positive, each of those actions is reinforced. If it is negative, they are punished. Reinforcement and punishment are achieved by backpropagating error signals associated with the network's activation in that situation and a training example derived from the training signal. For example, reinforcement on a *change left* decision would create a training example of the previous input paired with the target output (0.0, 1.0, 0.0). The targets of *stay center* and *change right* are 0.0 and change left is 1.0. Using the standard backpropagation procedure, the weights are updated based on this training example and the resulting network is more likely to choose *change left* in a similar situation. A negative training signal in the previous example would generate a target output of (1.0, 0.0, 1.0), and the resulting network would be less likely to choose *change left* in similar situations.

The learning strategy is somewhat similar to the temporal difference methods for reinforcement learning (Sutton, 1988), in that updates are made based on the performance differences over successive time periods. However, temporal difference methods treat performance differences as prediction errors from which they can learn to predict future rewards. Our local learning component uses the differences to determine whether to reinforce or penalize specific decisions. A temporal difference method could also be used as a local learning component in our framework, and we expect to evaluate this approach in the near future.

Experimental Evaluation

Intelligent lane selection appears to offer important advantages for traffic control and our learning approach appears to be a plausible methodology to generate the selection strategies. In this section, we test these hypotheses in a simulated traffic environment.

A Simulated Traffic Environment

To evaluate intelligent lane selection, we developed a simulator to model traffic on a highway. For each car, the simulator updates the continuous values of position, velocity, and acceleration at one second intervals. The acceleration and deceleration functions were set by visualizing traffic performance under different conditions and represent our best estimate of the behavior of actual drivers. Acceleration (A) is adjusted based on the equation $A(s) = 10s^{-0.5}$, where s represents the current speed in miles per hour (m/h).

Deceleration occurs at the rate of -2.0 m/h per second if the difference in speed from the immediate preceding car is greater than twice the number of seconds separating the two cars. In other words, if a car approaches a slower car, the deceleration point is in proportion to the difference in speed and the distance between the cars. If there is a large difference in speed, cars will decelerate sooner than if the speed differences are small. If the gap closes to two seconds, the speed is matched instantaneously. Lane changes are only allowed if the change maintains a two-second gap between preceding and following cars.

The simulated roadway is 3.3 miles long, but the top of each lane "wraps around" to the bottom, creating an infinite stretch of roadway. The simulator was designed as a tool to efficiently evaluate different lane-selection strategies and thus makes several simplifying assumptions about traffic dynamics. The primary assumptions in the current model are that:

- cars are the same size;
- cars use the same acceleration rules;
- cars accelerate to and maintain their desired speed if there are no slower, preceding cars;

Although these assumptions do not hold for real traffic, they are also not crucial to evaluate the merits of intelligent lane selection. Removing these assumptions unnecessarily complicates the model, which creates unacceptable run times for exploratory research. In future work, we will expand our experiments to a more realistic simulator such as SmartPATH (Eskafi, 1996).

During training, the learning system uses the traffic simulator to evaluate candidate lane-selection strategies. Each evaluation or *trial* lasts 400 simulated seconds and begins with a random dispersement of 200 cars over three lanes on the 3.3 mile roadway. Desired speeds are selected randomly from a normal distribution with mean 60 m/h and standard deviation 8 m/h. In each trial, the percentage of smart cars is random with a minimum percentage of 5%. All other cars follow the selfish lane selection strategy.

Each training run begins with a population of 75 random lane selection strategies and 25 seeded strategies, which are modified by SANE and the local learning

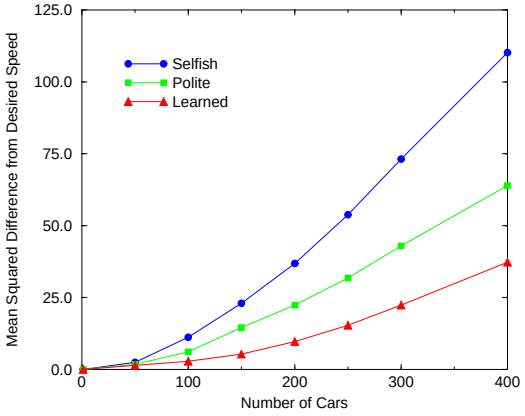


Figure 4: Speed error using different lane selection strategies under different traffic densities.

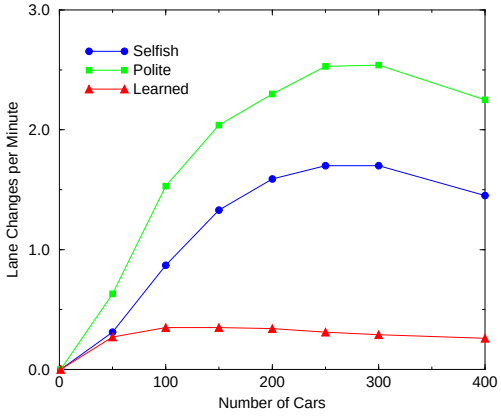


Figure 5: Average number of lane changes using different lane selection strategies under different traffic densities.

module over 30 simulated driving hours. SANE keeps track of the best strategy found so far based on its performance over a trial. When a better strategy is found, it is saved to a file for later testing. The saved strategies are each tested over ten 2000-second trials and the best is considered the final strategy of the training run.

Experiment 1: Traffic Densities

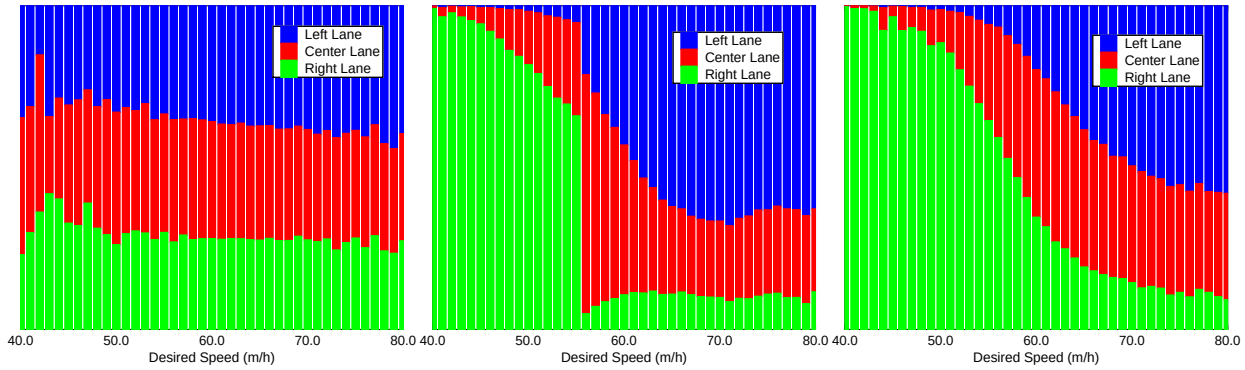
The first study compares the performance of traffic under different traffic densities using three different lane-selection schemes: a selfish strategy, a polite strategy, and the learned strategy. The selfish and polite strategies operate as described previously in this paper. The learned strategy is the best strategy from the five training runs. Here we are not interested in the aggregate performance over several learning runs, but rather in the performance of a single learned strategy that could be used in traffic. Experiments in (Tech. report reference omitted to ensure anonymity) more thoroughly evaluate the learning system and present learning curves averaged over all training runs.

Strategies were tested over car densities of 50 to 400 cars per 3.3 miles and performance was measured over 20 simulations at each density. In this experiment, all cars on the highway in a given condition employed the same strategy.

Figure 4 shows the error in driving speed for the selfish, polite, and learned strategy under different traffic densities. The error is computed from the first term in equation 1 and represents the average squared difference between actual speeds and desired speeds in m/h. The figure shows the clear advantage of the learned strategy. In sparse traffic (50-100 cars), the performance of the three strategies is comparable; however, in more dense traffic, the learned strategy produces significantly lower divergence from desired speeds. At a density of 200 cars, the learned strategy incurs only a quarter of the error of the selfish strategy and less than half the error of the polite strategy. The selfish strategy error grows faster in dense traffic than the polite and learned strategies, because of the many bottlenecks generated by the unyielding, slow drivers. The polite strategy solves many of these bottlenecks by moving slower drivers to the right, but still maintains a error of at least 20 m/h^2 over the learned strategy.

Figure 5 plots the average number of lane changes per car under each strategy. There is a large contrast in behavior between the polite and learned strategy. Even in very sparse traffic, the polite strategy produces over twice as many lane changes as the learned strategies. In heavy traffic, the polite strategy calls for almost nine times as many lane changes. The learned strategies reach a maximum lane change rate of 0.35 changes per minute, whereas the polite strategy reaches 1.53 lane changes per minute. The selfish strategy generates fewer lane changes than the polite strategy, since it does not have a yielding component; however, it still generates over five times as many lane changes as the learned strategy in denser traffic. Thus, compared to both the selfish and polite strategy, the learned strategy makes far fewer lane maneuvers, which should increase driver acceptance of intelligent lane selection and hopefully reduce accident rates.

Figure 6 provides a visualization of lane utilization under the different selection strategies for a density of 200 cars. Each graph represents an average over 20 simulations of the percentage of time a driver with a given desired speed spends in each lane. The selfish strategy, shown in Figure 6(a), assigns no lane bias to faster or slower drivers, and thus drivers at different speeds are spread across all three lanes fairly evenly. The polite strategy, in Figure 6(b), does bias slow drivers towards the right lane and fast drivers towards the left lane, but does so with a rigid partition at 55 m/h. Thus, a car with a desired speed of 54 m/h behaves quite differently than a car with a desired speed of 56 m/h. This partition comes from the polite



(a) (b) (c)

Figure 6: Utility of lanes with respect to desired speeds for the (a) selfish, (b) polite, and (c) learned strategies. The graph shows the percentage of time that cars drive in the left, center, and middle lanes as a function of desired speeds. These tests used a traffic density of 200 cars per 3.3 miles.

	Left Lane	Center Lane	Right Lane
Selfish	0.35	0.35	0.30
Polite	0.35	0.26	0.39
Learned	0.25	0.27	0.48

Table 1: The distribution of traffic for the three lane selection strategies.

rule that moves cars traveling slower than 55 m/h to the right lane. The learned strategy, in Figure 6(c), produces a much smoother lane utilization bias.

Another contrast between the three strategies lies in the overall utilization of the three lanes across all speeds. Table 1 shows the overall lane distribution for all cars. The learned strategy has a significant bias towards the right lane and places almost half of the cars there. This organization seems reasonable and quite effective since slower cars encounter fewer slower preceding cars and should operate more efficiently in higher traffic density than faster cars. The learned lane-selection strategy essentially moves half of the traffic to the right lane and uses the middle and left lanes to organize the faster traffic. It is also important to note from Figure 6 that the faster cars do appear in the right lane, but the slower cars never appear in the left lane. The likely reason is that a slow car in the left lane causes large disruptions to traffic flow, whereas a fast car in the right lane will normally only disrupt its own performance.

Experiment 2: Mixing Strategies

The second experiment evaluated the learned strategy in the presence of selfish cars. The aim was to examine the robustness of the smart car's group behavior to cars that do not follow the same rules. We were interested in how quickly the learned strategy degrades as

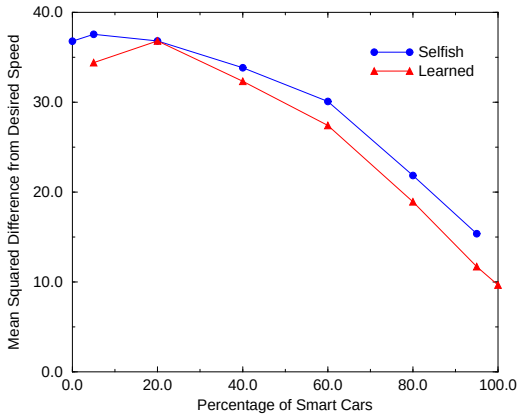


Figure 7: The average squared error of smart and selfish cars under different smart car distributions. These tests used 200 cars.

selfish drivers are added and how many smart cars are necessary to make cooperative behavior worthwhile.

Figure 7 shows the error in driving speeds under different smart car distributions. The figure plots the speed error for both the smart cars and the selfish cars and illustrates how the performance of both increases with the introduction of more smart cars. The figure shows that, even at distributions as low as 5% smart cars, there is incentive to cooperate. At 100% selfish traffic, cars average a 36.80 m/h^2 driving error, while at 95% selfish traffic the error drops to 34.40 m/h^2. While this is not a substantial improvement it demonstrates that very few smart cars are necessary to generate an increase in traffic performance. Moreover, performance improves steadily as more cars cooperate, which provides further motivation to drive cooperatively. Finally, at 100% smart cars the average speed

error drops to 9.66 m/h^2, which is approximately one fourth of the error when all traffic is selfish.

Related Work

Intelligent lane selection appears to be a novel approach to traffic management that has received almost no attention in the traffic management literature. After a lengthy literature search and several email inquiries, we have found only one project with similar goals. Carrara and Morello have proposed a system called DOMINC that employs cooperative driving techniques to increase traffic efficiency.[1] The main objective of the DOMINC project is to explore the benefits in traffic efficiency, comfort, and safety of cooperative driving. The project's vision is thus very close to our formulation of car-centered traffic management. However, the paper that we have only describes the potential benefits and does not propose a specific methodology for cooperative driving.

There are a number of systems designed to learn lane selection for a single-agent, non-cooperative system. McCallum (1996) used reinforcement learning to train a driving agent to weave around traffic, a task that he calls "New York driving". Sukthankar, Baluja, and Hancock (1997) used an approach similar to evolutionary algorithms to form a voting scheme that determines the appropriate driving lane and speed for a single car. Finally, the Bayesian Automated Taxi (BAT) project, an attempt to build a fully automated vehicle that can drive in normal traffic (Forbes, Huang, Kanazawa, & Russell, 1995), will eventually contain a module for lane selection. Each of these systems were designed to maximize the performance of a single vehicle and do not form cooperative controllers. Our approach is directed at the global traffic management problem, where cooperation is important.

Summary and Conclusions

Coordination of local car behaviors is a novel approach to traffic management that poses a challenging problem to both artificial intelligence and machine learning. In this paper, we proposed one formulation of this problem: intelligent lane selection to maintain desired driving speeds and reduce lane changes. Given only information on the local traffic patterns and the desired speed, cars can coordinate local lane changes to let faster traffic pass through while still allowing slower traffic to maintain desired speeds.

We described and evaluated an approach that uses supervised and reinforcement learning to generate the lane-selection strategies through trial and error interactions with the traffic environment. Compared to both a selfish strategy and the standard "yield to the right" strategy, the smart cars maintained speeds closer to the desired speeds of their drivers while making fewer lane changes. Additionally, intelligent lane selection was shown robust in the presence of selfish drivers. Traffic performance improves even when as few as five percent of the cars cooperate. Future work will explore more realistic traffic and driver models, as well as variations on the coordination task.

[1] The paper that we have does not include a reference.

References

Carrara, M., & Morello, E. Advanced control strategies and methods for motorway of the future. In *The Drive Project DOMINC: New Concepts and Research Under Way.*

Dietterich, T. G. (1990). Exploratory research in machine learning. *Machine Learning, 5*, 5–9.

Eskafi, F. (1996). *Modeling and Simulation of the Automated Highway System.* Ph.D. thesis, Department of EECS, The University of California at Berkeley.

Forbes, J., Huang, T., Kanazawa, K., & Russell, S. (1995). The BATmobile: Toards a bayesian automated taxi. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)* Montreal, CA.

Gilmore, J. F., Elibiary, K. J., & Forbes, H. C. (1994). Knowledge-based advanced traffic management system. In *Proceedings of IVHS America* Atlanta, GA.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA.

Kagolanu, K., Fink, R., Smartt, H., Powell, R., & Larson, E. (1995). An intelligent traffic controller. In *Proceedings of the Second World Congress on Intelligent Transport Systems*, pp. 259–264 Yokohama, Japan.

McCallum, A. K. (1996). Learning to use selective attention and short-term memory in sequential tasks. In *Proceedings of Fourth International Conference on Simulation of Adaptive Behavior*, pp. 315–324 Cape Cod, MA.

Moriarty, D. E. (1997). *Symbiotic Evolution of Neural Networks in Sequential Decision Tasks.* Ph.D. thesis, Department of Computer Sciences, The University of Texas at Austin.

Moriarty, D. E., & Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning, 22*, 11–32.

Pooran, F. J., Tarnoff, P. J., & Kalaputapu, R. (1996). RT-TRACS: Development of the real-time control logic. In *Proceedings of the 1996 Annual Meeting of ITS America*, pp. 422–430 Houston, Tx.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E., & McClelland, J. L. (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, pp. 318–362. MIT Press, Cambridge, MA.

Sukthankar, R., Baluja, S., & Hancock, J. (1997). Evolving an intelligent vehicle for tactical reasoning in traffic. In *Proceedings of the IEEE International Conference on Robotics and Automation.*

Sutton, R. S. (1988). Learning to predict by the methods of temproal differences. *Machine Learning, 3*, 9–44.