

Learning Distributed Strategies for Traffic Control

David E. Moriarty, Simon Handley, and Pat Langley

Daimler-Benz Research and Technology Center
1510 Page Mill Road, Palo Alto, CA 94304
{moriarty,handley,langley}@rtna.daimlerbenz.com

Abstract

In this paper, we cast the problem of managing traffic flow in terms of a distributed collection of independent agents that adapt to their environment. We describe an evolutionary algorithm that learns strategies for lane selection, using local information, on a simulated highway that contains hundreds of agents. Experimental studies suggest that the learned controllers lead to better traffic flow than ones constructed manually, and that the learned controllers are robust with respect to blocked lanes and changes in the number of lanes on the highway.

1. Introduction

In recent years, there has been growing interest in the distributed behavior of large populations of independent agents, and in the ability of such agents to adapt not only to their environment but to each other's behavior. Most research in this area has focused on agent populations designed to mimic those that occur in the natural world, such as colonies of ants and schools of fish. However, the artificial urban environment created by humans contains another important example of distributed agent behavior: automobile traffic on roads and highways.

The traffic domain has much to recommend it as a fertile source of research problems. Clearly, each agent (a driver and his vehicle) has independent control of its actions, but its behavior must take into account physical constraints, such as staying on the road and avoiding collisions, and the behavior of many other agents. Another advantage is that there exist clear criteria for evaluation, such as maximizing traffic flow and minimizing lane changes. In addition, the domain supports complex behaviors of the overall system, such as traffic jams, even though the individual agents are relatively simple. Also, most researchers have personal experience in traffic environments and, presumably, have good intuitions about reasonable strategies and behaviors. Finally, progress in this area could lead to improvements in actual traffic conditions and thus increase the quality of life for drivers.

In this paper, we describe research on distributed adaptation in the traffic domain. Following Moriarty and Langley (1998), we formulate the problem of traffic management from a distributed, car-centered perspective and the task of improving this process in terms of distributed

machine learning. We assume that each agent receives local information about the vehicles that immediately surround it, including their location and speed, and that the agent determines which lane to drive in but not the vehicle's speed. The aim of learning, and thus our measure of performance, is not the behavior of individual traffic agents but rather the behavior of the traffic system as a whole. To this end, we want the learning module to develop a control strategy for lane selection that considers not only the maintenance of each car's desired speed, but that takes into account how its selection will affect speeds of other cars. For instance, we would like the cars to organize themselves into a cooperative system that lets the fast drivers pass through, while still letting the slow drivers maintain their speeds.

We begin by presenting our formulation of the traffic control task, and its associated learning problem, in more detail. We next describe the inputs and outputs of our reactive vehicle controllers, along with our genetic approach to learning distributed control strategies. After this, we characterize our simulated traffic environment and report experimental studies of the system under a variety of conditions. Elsewhere (Moriarty & Langley, 1998), we showed that our approach learns controllers that are robust to changes in the proportion of learned to 'selfish' cars on the highway and to changes in traffic density. Here we focus instead on the system's ability to generalize to situations that involve blocked lanes and different numbers of lanes from those used in training. We close the paper with a discussion of related work on distributed learning and our plans for future research.

2. Distributed Traffic Control

Our approach to traffic management involves a reformulation of the problem into a distributed artificial intelligence task, in which cars coordinate lane changes to maintain desired speeds and reduce total lane maneuvers. We make no assumptions about the level of automation of the cars. Lane selection information could be provided to the driver, who completes the maneuver, or to a regulation controller in an automated car (Pomerleau, 1995; Varaiya, 1993). Here we consider in more detail our formulation of the performance and learning tasks.

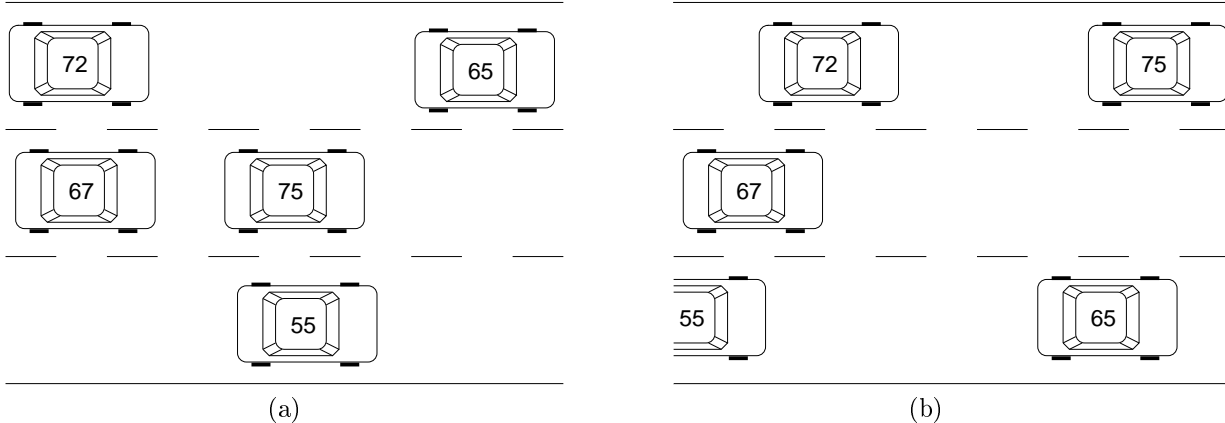


Figure 1 (a) An example traffic situation in which the traffic flows from left to right and the number on each car shows the car’s speed. (b) Traffic after reorganization in which car 75 and 65 swap lanes, followed by another lane change by car 65, so that all cars can maintain their desired speeds.

2.1 Definition of the Problem

Most work on advanced traffic management views cars as tokens that follow simple, selfish rules of behavior. These management systems affect the flow of the car tokens by controlling external, fixed-position devices such as traffic signals, ramp meters, speed limits, and dynamic lanes. Surprisingly, little research has addressed how the cars themselves might sense and intelligently affect traffic dynamics.¹

Our view is that cars are not blind tokens, but rather can sense their environment and act cooperatively to achieve desired global behavior. More specifically, cars can learn to organize themselves by traffic lanes to increase overall traffic throughput, reduce the average number of lane changes, and maintain the desired speeds of drivers. Car-centered control, specifically lane selection, should therefore complement existing traffic management efforts by providing better behavior between traffic signals.

Figure 1(a) illustrates a situation in which lane coordination is beneficial. The figure shows five cars along with their speeds, which we will use as identifiers. Car 72 is quickly approaching car 65 and will be 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. An efficient solution to this problem is for car 75 and car 65 to immediately swap lanes, followed by car 65 moving into the bottom lane, as shown in Figure 1(b). This maneuver ensures that no speeds are reduced and no throughput is lost.

We recast the traffic management task as a problem in distributed artificial intelligence, where each car represents an individual agent in a multi-agent system. Cars

act on their world (the highway) by selecting appropriate lanes to drive in. They interact with other cars by competing for resources (the spaces or slots 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 performance 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.

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 measure P for a set of cars C contains two terms, one that penalizes deviations in speed and one that penalizes lane changes,

$$P(C) = \frac{\sum_{t=1}^T \sum_{i=1}^N (S_{it}^a - S_{it}^d)^2}{TN} + 4 \times 60 \times \frac{\sum_{i=1}^N L_i}{TN}, \quad (1)$$

where T is the total number of 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 first constant, 4, is a weighting factor, whereas the second, 60, converts the lane changes per second into lane changes per minute. The goal is to minimize the difference between actual speeds and desired speeds, modulated by the number of lane changes, averaged over several time steps and over all learned cars on the road. 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 gives the same

¹ One exception is the work of Carrara and Morello in the DOMINC project.

average. Squaring the error from desired speed gives a higher evaluation to the more consistent strategy.

The problem is thus to find a lane-changing strategy that minimizes 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*, since they maximize the local performance of their respective car. We are interested in *smart* strategies that maximize the aggregate performance of traffic through cooperative lane-selection strategies.

2.2 Communication and Coordination

The previous section defined the problem of car-centered traffic management, but left open some important issues about the level of communication between cars and the knowledge available about other cars’ decisions and states. The multi-agent literature is often divided on these matters, and we feel that it is not central to the problem definition. Still, we should 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 agents can perceive limited state information of surrounding cars, such as their relative speeds. The system could sense this information using radar or receive it directly from other cars via radio waves or the Internet. Agents should also sense which surrounding cars are cooperative and which are selfish. Again, the system could infer cooperation from driver behavior or direct communication.

Figure 2 illustrates the input for an agent 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 distant traffic. For example, it may prove beneficial to receive not only relative speeds of individual cars in the immediate vicinity, but also relative speeds of groups of cars in farther ranges.

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

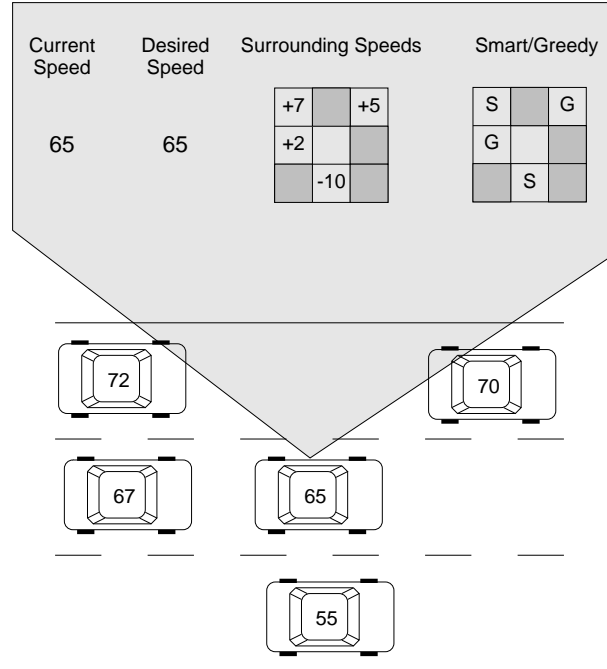


Figure 2 An illustration of the input to each agent. The shaded region shows the current input information for the middle car. The agent has access to its current speed, its desired speed, the relative speeds of surrounding traffic, and whether other cars are smart or selfish.

lanes or on knowledge of the current driving lane. Thus, controllers that learn on a three-lane highway should, at least in principle, generalize to greater or fewer lanes.

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 being selected as the car’s next action. For a recommendation to be valid, there must be a lane available in the specified direction. For a recommendation to be safe, there must not be a car in the same longitudinal position in the new lane. It is always safe to remain in the current lane. The system could also incorporate other safety assurances, such as detecting whether a lane change produces an unsafe spacing between cars in the new lane. For example, one might specify that a slow car should not move in front of a fast car even if there is immediate space for it in the fast car’s lane, since the fast car will likely close that space during the span of the lane change.

The higher-level safety and validation process relieves the controller of the overhead in deciding which lanes are safe and centers the control problem on lane selection. In other words, by removing the problem of validation and safety, the controller can focus on and more easily learn to rank lanes. This approach is analogous to separating the identification of legal moves from the selection of desirable moves in game playing.

Another important issue concerns support for individual differences among drivers. Clearly, different drivers should be able to select lanes differently. Slower drivers will normally (but not always) use lane selection to open up lanes for faster traffic, whereas faster drivers will select lanes to get through slower traffic. Average-speed drivers will employ elements of both strategies. At issue is how to represent and implement the different types of strategies.

One approach is to maintain an explicit control policy for each type of driver. For example, fast drivers would utilize a fast lane-selection strategy and slow drivers a slow lane-selection strategy. A disadvantage of this approach is that it requires *a priori* knowledge of the number of driver types and the boundaries that separate them. Also, it does not provide a smooth transition between styles of driving. A driver on a boundary would be forced into one of the two surrounding strategies instead of an interpolation between the two.

A better approach is to parameterize the driving style and use it as input to a single control policy. Each car would contain the same control policy, but since it receives driving style as input, it behaves differently for different types of drivers. In this case, driving style is simply the desired speed. No *a priori* decisions are necessary regarding the number of lane-selection strategies or their boundaries. Moreover, since the different strategies are keyed to a continuous input (desired speed), there can be smooth transitions and interpolations between different lane-selection strategies.

2.3 Approaches to Intelligent Lane Selection

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 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.

Another solution is to apply machine learning to develop intelligent controllers through direct experience with the domain. A learning algorithm would modify the controller based on good and bad experiences in the problem space. This approach frees us from the task of acquiring and encoding expert domain knowledge, since it discovers examples of good and bad decisions through direct experience. Moreover, the controllers are not necessarily fixed and could continue to learn and adapt with new experiences.

The lane-selection problem appears out of reach of the more standard, *supervised* machine learning methods (e.g., Quinlan, 1986; Rumelhart, Hinton, & Williams, 1986). In supervised learning, control policies are formed

from examples of correct behavior. In the case of intelligent lane selection, supervised learning requires demonstrations of good and bad lane selections. Without expert domain knowledge, it is difficult to generate these examples. In some control problems, supervised learning is used to mimic the behavior of people (e.g., Pomerleau, 1992; Sammut, Hurst, Kedzier, & Michie, 1992). For intelligent lane selection, however, this is exactly what we do not want to model. We believe that most drivers do not select lanes intelligently, but are rather more selfish in nature. Thus, it seems misguided to use real driver behaviors as a basis for learning cooperative lane selection.

A more flexible machine learning approach that is capable of learning from general rewards instead of behavioral examples has been termed *reinforcement learning*. The rewards provide only a general measure of proficiency over the task and do not explicitly direct the learner toward any course of action. The learner adjusts its actions through trial and error interactions with the environment to maximize the reward signal. In the lane-selection problem, agents receive the rewards defined by equation 1 at specific time steps. In response, they adjust their lane-selection strategies using some reinforcement learning algorithm to maximize the reward function.

The literature includes two main types of approach to reinforcement learning. One class of methods learn through calculation of temporal differences (Sutton, 1988; Watkins & Dayan, 1992; Kaelbling, Littman, & Moore, 1996) over rewards, which lets them acquire mappings from state-action pairs onto expected values. Another class of methods search more directly through the space of control policies (Grefenstette, Ramsey, & Schultz, 1990; Holland & Reitman, 1978; Moriarty & Miikkulainen, 1996; Whitley, Dominic, Das, & Anderson, 1993; Wilson, 1994), often using evolutionary algorithms to this end. Our approach, to which we now turn, relies on an evolutionary algorithm as the primary mechanism for reinforcement learning, but it also incorporates a technique similar to temporal-difference learning to handle smaller strategy refinements.

2.4 Machine Learning for Lane Selection

Elsewhere (Moriarty & Langley, 1998), we have described our distributed learning system in detail, so here we only review its main components. The system represents its control knowledge as a feedforward neural network with one hidden layer, as depicted in Figure 3. This network includes 16 input units, 12 hidden units, and three output units, with full connections between adjacent levels. The input nodes correspond to information about the car's current and desired speeds, as well as the speeds of surrounding vehicles, whereas the output nodes specify whether to stay in the current lane, to move left, or to move right. On each time step, the controller uses the input values to compute activations for each output node, then selects the action with the highest activation.

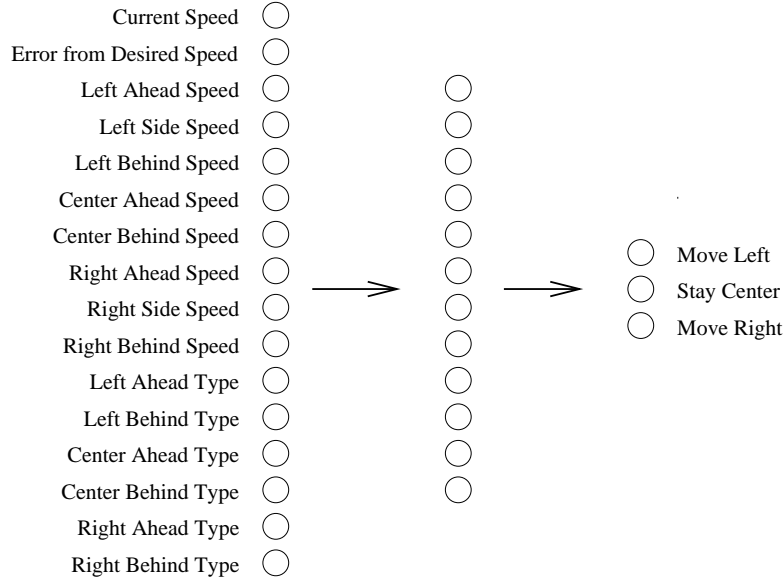


Figure 3 The input and outputs to the neural network for lane selection.

The learning system relies on three interrelated modules to determine the weights on the network's links, and thus to acquire robust controllers. The first component is SANE (Moriarty & Miikkulainen, 1996; Moriarty, 1997), which carries out genetic search through the space of feedforward networks, given a network architecture, by operating at two distinct levels. At one level, the module retains a population of complete controllers, each defined as a collection of hidden-layer neurons. The second-level population consists of these individual neurons, each of which specify the weights on their input and output links. Each member of the neuron population can appear in zero or more members of the controller population.

SANE assign fitness to each candidate controller by converting its encoding (stored as bit strings) into a complete neural network, then using that network in a simulated traffic environment for 400 seconds. The system determines the fitness of a given controller by applying equation 1 to traces of the traffic behavior. SANE also assigns fitness to each candidate neuron based on the fitnesses of the controllers to which it contributes. The algorithm selectively applies genetic operators, such as crossover, on members of each population to generate new members, repeating the evaluation and generation process many times.

The second learning module is responsible for seeding the initial populations. Rather than starting SANE with random populations of controllers and neurons, the system initializes them with candidates that are likely to be useful. The module accomplishes this feat by collecting behavioral traces of the hand-written 'polite' controller that we described earlier. These provide training cases that take the form of sensor-action pairs, which the sys-

tem passes to a supervised algorithm (backpropagation) to learn weights that approximate the polite controller. The module repeats this process a number of times to generate a collection of controllers and neurons that form 25% of the two initial populations.

As noted above, the SANE module evaluates the fitness of candidate controllers by running them in the simulator for fixed time periods. Because the reward signals generated by this scheme are infrequent, we added a third learning module that relies on more immediate feedback. After every ten simulated seconds, this component checks to determine if the overall traffic performance has changed since the last measurement. If performance has improved substantially, it labels all actions taken during this period as desirable; if performance has worsened, it labels all invoked actions as undesirable. In either case, the module passes these actions (and their associated sensory inputs) to the backpropagation algorithm, which alters the controllers weights to either encourage or discourage their use.

Recall that the system's reward signal is based on equation 1, which assumes global information about overall traffic behavior. Clearly, such information is not currently available to actual cars or their drivers, but we predict this will change as vehicles come to include positioning devices and gain access to the Internet, which will let them report their position and speed to a central facility. For now, we have been forced to rely on a simulated traffic domain, which has also encouraged us to use an offline training regimen to collect accurate statistics over extended runs. However, the basic approach also lends itself to online learning, though we expect the learning rate would decrease in this scenario.

3. Experimental Evaluation

Our approach to distributed learning appears to offer a viable method for acquiring lane-selection strategies and thus improving overall traffic performance. However, whether the method works in practice is an empirical question, and in this section we report experimental studies of the system’s adaptive behavior.

3.1 A Simulated Traffic Environment

To evaluate traffic management through 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. We adjust acceleration (A) using 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 proportional 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. The simulator allows lane changes only if the change maintains a two-second gap between leading and following cars.

The simulated roadway is 3.3 miles long, but the top of each lane “wraps around” toroidally to the bottom, creating an infinite stretch of roadway. We designed the simulator as a tool to efficiently evaluate different lane-selection strategies, and thus it makes several assumptions about traffic dynamics. The current model makes five primary assumptions:

- all cars are the same size;
- all cars use the same acceleration rules;
- cars accelerate to and maintain their desired speed if there are no slower cars directly ahead;
- lane changes are instantaneous; and
- there are no curves, hills, on ramps, or exit ramps.

Although none of these assumptions hold for real-world traffic, they do not appear crucial for evaluating the merits of intelligent lane selection, and removing them unnecessarily complicates the model. In future work, however, we hope to expand our experiments to more realistic simulators 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 dispersment 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 randomly selected from a uniform distribution with a minimum percentage of 5%. All other cars follow the selfish lane-selection strategy outlined in Section 2.1.

To simulate congestion caused by lane closures and merging, we blocked portions of either the far right or far left lanes during training. Lane closures last for one mile and only one closure exists at any given time. There is an equal probability that the far right or far left lane will be blocked. A lane-selection strategy perceives a blocked lane as a car with a speed of zero.

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

We developed two hand-written controllers for use as benchmarks and to provide the simulated environment. The *polite* controllers follow four rules:

- If your desired speed is 55 m/h or less and the right lane is open, then change lanes to the 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 to the 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 to the left.
- In the previous situation, if the left lane was not open but the right lane is open, then change to the right.

We based these rules on our interpretation of the “slower traffic yield to the right” signs posted on the highways. The selfish strategy described earlier uses only the last two rules.

3.2 Evaluation of Intelligent Lane Selection

Our earlier studies (Moriarty & Langley, 1998) evaluated the learned controllers’ behavior as we varied the density of traffic and the ratio of learned to selfish controllers. The results, which showed reasonable behavior over a wide range of densities and ratios, encouraged us to carry out additional studies to further test the adaptive nature of the learned controllers.

We designed our first experiment to evaluate the polite, selfish, and learned strategies in the presence of lane closures. Recall that, during training, we closed one mile of either the far left or far right lane and the location of the closure changed every 500 simulated seconds. We replicated lane closures in testing to determine each

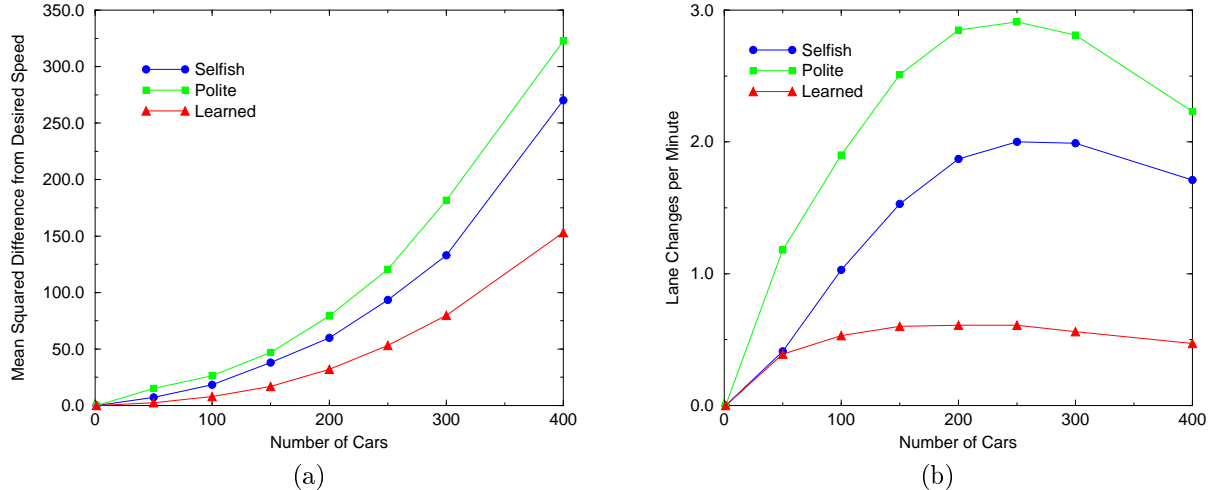


Figure 4 Traffic performance when portions of the lanes were blocked.

strategy’s ability to handle high congestion areas created by the merging traffic. The degree of merging in these tests is extreme (a blocked lane every 13 miles), to fully test the robustness of the three strategies.

Figure 4 plots the mean squared error in desired speeds and the average number of lane changes with closed lanes. Surprisingly, the polite strategy performed worse than the selfish one when lanes were blocked, which differed from our earlier results with no closures. Figure 4(a) shows that, under a high degree of merging, it is better to act greedily than politely. The large errors that the polite strategy incurs come when portions of the rightmost lane are closed. Since the polite strategy directs all of its slow drivers into the right lane, it becomes difficult to merge them back into the two faster lanes when this lane is blocked. This difficulty causes large bottlenecks in the right lane and creates high errors in desired speed. Since the selfish strategy assigns no lane bias based on driving speed, it is less affected by right lane closures.

Although the learned strategy also directs its slower drivers to the right lane, its response to bottlenecks is even more robust than the selfish scheme. Under the learned strategy, faster drivers in the center and left lanes maneuver to let slower drivers merge more easily, which eases congestion. These seemingly altruistic behaviors were learned because reinforcement comes from the aggregate traffic performance. Additionally, the learned cars have relative speed sensors that can detect slow speeds in traffic ahead. Thus, the learned strategy can merge the cars much earlier than the polite strategy, which does not begin to merge until a slow car or closed lane forces it to decelerate. Overall, the learned strategy incurs substantially lower driving errors and performs only a fraction of the lane change maneuvers as the other two strategies.

The second experiment evaluated the learned strategy’s ability to adapt to four-lane highways. As noted in Section 2.2, we designed the controller input to ignore the car’s actual lane, and the output to reflect only whether the left or right lane is better than the current one. Thus, in principle an effective strategy learned only on a three-lane highway should perform well on four lanes. Since the learning system only experienced three-lane highways in training, this experiment serves as another test of the learned strategy’s adaptability.

Figure 5 plots the error in driving speed and average number of lane changes using four lanes of traffic. Since there is more lane capacity, we need up to 600 cars in this study. The figure shows that the learned strategy achieves the same performance gains over the polite and selfish strategies in four lanes of traffic as it does in three lanes. In dense traffic, the learned strategy incurs one third to one quarter of the driving speed error for the selfish strategy and one half of the error for the polite strategy. As with three lanes of traffic, the polite and selfish strategies make substantially more lane change maneuvers than the learned controller.

Figure 6 provides a visualization of the lane utilization for the three strategies with four traffic lanes using 300 cars. This shows that the selfish strategy assigns no lane bias based on driving speed, whereas the polite strategy exhibits a sharp transition between slow and fast driving styles. The graph for the learned strategy is very similar to its three-lane counterpart, giving further evidence that it generalizes to more than three lanes.

With only three lanes, the learned strategy encumbers the right lane with many slow drivers and uses the other lanes to organize the faster drivers. Under four lanes of traffic, however, the fastest drivers are placed more consistently in the leftmost lane than under three lanes. For example, drivers with desired speeds of 80

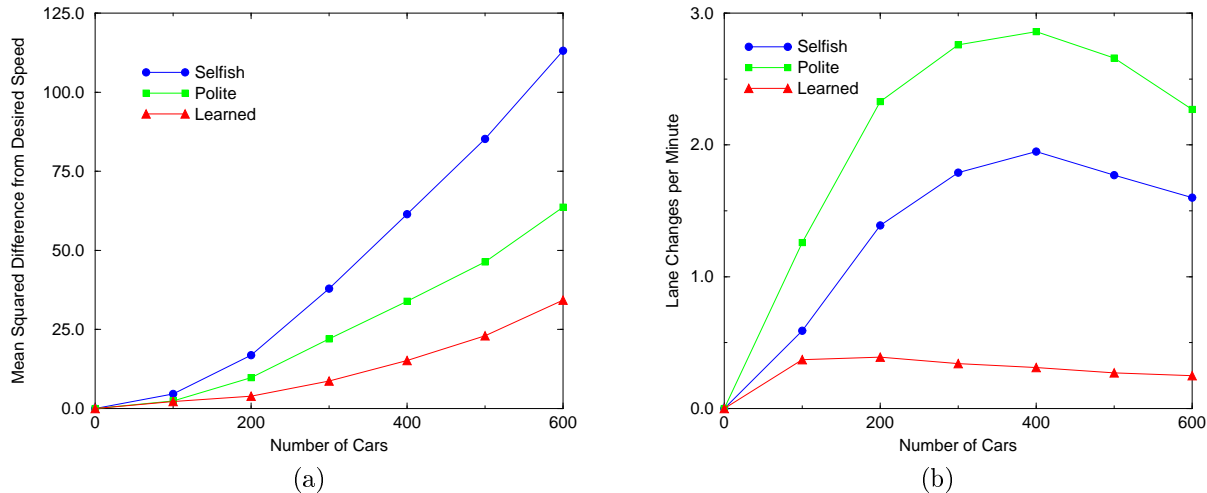


Figure 5 Traffic performance with four driving lanes; no lanes were blocked in this condition.

m/h drive in the left lane 83% of the time with four lanes of traffic, compared to only 57% of the time with three lanes. Most of the traffic organization occurs in the middle two lanes, with the middle-speed drivers. This strategy seems reasonable, since the middle-speed drivers make two different types of lane changes: passing slow cars and yielding to fast cars, and therefore must reorganize more frequently.

4. Related and Future Work

One can roughly divide research on communities of learning agents into two broad categories. The first, often called *multi-agent learning*, refers to situations in which the agents have shared goals and thus cooperate, either explicitly or implicitly, to achieve those goals. Examples of this approach include Schultz, Grefenstette, and Adams' (1996) work on multi-robot herding behavior, Mataric's (1994) efforts on foraging, in which four robots acquire social rules that reduce disruption, Tan's (1993) studies of reinforcement learning among predators cooperating to track down prey, and Sen and Sekaran's (1998) use of reinforcement learning to improve two-agent coordination in block pushing. Stone and Veloso (1997) present an extensive review of work on multi-agent learning, including their own results on soccer playing, so we will not try to be exhaustive here.

Another category focuses on situations that involve many agents, typically more than in multi-agent settings, each of which pursues its own goals. Research on such *distributed learning* seems less common than multi-agent work, but it also bears a closer relation to our own approach. Perhaps the best-known effort of this sort revolves around Holland's (1996) ECHO, a simulation framework designed to study the behavior of complex biological systems, such as the interaction of plants, herbivores, and carnivores in an ecosystem (Schmitz & Booth,

1997). Schoonderwoerd, Holland, and Bruten (1997) use distributed agents to balance loads in telecommunications networks, but learning occurs only in the sense that agents lay down ant-like trails to improve performance. Grand, Cliff, and Malhotra's (1997) work on CREATURES is more akin to our own work, with independent agents that exist in a simulated environment, receive rewards, and change their behaviors with experience.

There does exist some work on machine learning for traffic control, but this has focused on learning for individual driving agents. For example, Sukthankar et al. (1996) use reinforcement learning to acquire control strategies for a vehicle that operates on a simulated highway among other cars controlled by hand-crafted strategies. Similarly, McCallum (1996) reports a system that uses reinforcement learning to acquire a single-agent controller for 'New York driving', which involves weaving around slower traffic. There also exists a substantial literature on more traditional approaches to traffic management that typically involve more centralized control, which we have reviewed in a separate paper (Moriarty & Langley, 1998).

Although we have found no other work on distributed learning in the traffic domain, we remain excited about its potential as a fertile research testbed. In future work, we plan to improve our traffic simulator to include durative lane changes, entrance ramps, and exit ramps. We believe these additions will produce more realistic congestion patterns and thus increase the need for intelligent lane selection. The revised simulator will also let the vehicles increase or decrease their speed, which should provide further improvements in traffic flow, at least for learned controllers.

In the longer term, we envision an extended simulator that supports a network of interconnected highways. Each car would be given a destination and, to the extent

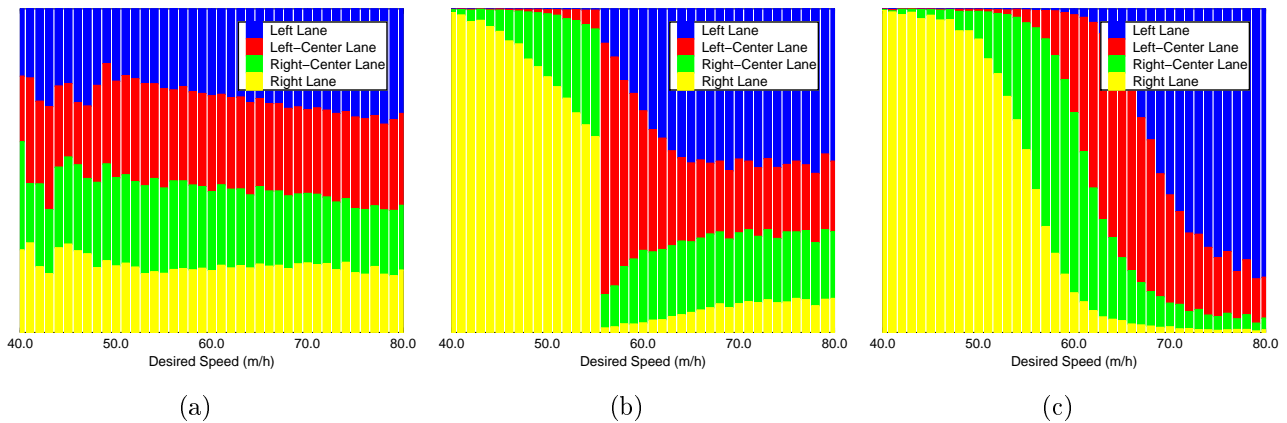


Figure 6 Utility of lanes with respect to desired speeds for the (a) selfish, (b) polite, and (c) learned strategies with four traffic lanes.

that multiple routes are available, the controller will select among routes just as it currently selects among lanes. This will require access to higher-level information about the distribution of cars on the various highways, so that the distributed controllers can select routes that avoid congestion. These variations on the task of distributed traffic management should provide a rich set of problems to drive our research in years to come.

However, our research methodology should remain much the same, in that we will construct systems that control traffic in a distributed manner and we will study those systems' adaptive behavior under a variety of experimental conditions. We invite other researchers to join us in our exploration of an intriguing domain that remains poorly understood despite its relevance to our everyday lives.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments and Dan Shapiro for his evaluation of traffic simulators.

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*.
- Eskafi, F. (1996). *Modeling and simulation of the automated highway system*. Ph.D. thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley.
- Grand, S., Cliff, D., & Malhotra, A. (1997). CREATURES: Artificial life autonomous software agents for home entertainment. In *Proceedings of the First International Conference on Autonomous Agents*, pp. 22–29. New York: ACM Press.
- Grefenstette, J. J., Ramsey, C. L., & Schultz, A. (1990). Learning sequential decision rules using simulation models and competition. *Machine Learning*, 5, 355–381.
- Holland, J. H. (1996). *Hidden order: How adaptation builds complexity*. Reading, MA: Addison-Wesley.
- Holland, J. H., & Reitman, J. S. (1978). Cognitive systems based on adaptive algorithms. In Waterman, D. A., & Hayes-Roth, F. (Eds.), *Pattern-directed inference systems*. New York: Academic Press.
- Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, 237–285.
- Matarić, M. J. (1994). Learning to behave socially. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pp. 453–462. Cambridge, MA: MIT Press.
- 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, University of Texas at Austin.
- Moriarty, D. E., & Langley, P. (1998). Learning cooperative lane selection strategies for highways. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence* Menlo Park, CA: AAAI Press.
- Moriarty, D. E., & Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22, 11–32.

- Pomerleau, D. (1995). RALPH: Rapidly adapting lateral position handler. In *Proceedings of the 1995 IEEE Symposium on Intelligent Vehicles*, pp. 506–511 Detroit, MI.
- Pomerleau, D. A. (1992). *Neural network perception for mobile robot guidance*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1, 81–106.
- 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*. Cambridge, MA: MIT Press.
- Sammut, C., Hurst, S., Kedzier, D., & Michie, D. (1992). Learning to fly. In *Proceedings of the Ninth International Workshop on Machine Learning*, pp. 385–393. San Francisco: Morgan Kaufmann.
- Schmitz, O., & Booth, G. (1997). Modelling food web complexity: The consequences of individual-based, spatially explicit behavioral ecology on trophic interactions. *Evolutionary Ecology*, 11, 379–398.
- Schoonderwoerd, R., Holland, O., & Bruten, J. (1997). Ant-like agents for load balancing in telecommunications networks. In *Proceedings of the First International Conference on Autonomous Agents*, pp. 209–216. New York: ACM Press.
- Schultz, A. C., Grefenstette, J. J., & Adams, W. (1996). RoboShepherd: Learning a complex behavior. In *Proceedings of RoboLearn-96: International Workshop for Learning in Autonomous Robots* Key West, FL.
- Sen, S., & Sekaran, M. (1998). Individual learning of coordination knowledge. *Journal of Experimental & Theoretical Artificial Intelligence*, 10.
- Stone, P., & Veloso, M. (1997). Multiagent systems: A survey from a machine learning perspective. Tech. rep. CMU-CS-97-193, School of Computer Science, Carnegie Mellon University.
- Sukthankar, R., Hancock, J., Baluja, S., Pomerleau, D., & Thorpe, C. (1996). Adaptive intelligent vehicle modules for tactical driving. In *Proceedings of the AAAI-96 Workshop on Intelligent Adaptive Agents*, pp. 13–22 Portland, OR. Also available at <http://www.cs.cmu.edu/~rahuls/Shiva/>.
- Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3, 9–44.
- Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 330–337. San Francisco: Morgan Kaufmann.
- Varaiya, P. (1993). Smart cars on smart roads: Problems of control. *IEEE Transactions on Automatic Control*, 38, 195–207.
- Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, 8, 279–292.
- Whitley, D., Dominic, S., Das, R., & Anderson, C. W. (1993). Genetic reinforcement learning for neuro-control problems. *Machine Learning*, 13, 259–284.
- Wilson, S. W. (1994). ZCS: A zeroth level classifier system. *Evolutionary Computation*, 2, 1–18.