

Distributed Learning of Lane-Selection Strategies for Traffic Management



Moriarty, D., & Langley, P. (1998)
(Technical Report 98-2). Daimler-Benz Research & Technology Center, Palo Alto, CA.

Presented by: Dave Kauchak
Department of Computer Science
University of California, San Diego
dkauchak@cs.ucsd.edu

Traffic Management



- Reduce congestion
- Increase throughput
- Reduce travel time
- Increase safety



Previous methods




- Traffic signal control
- Highway ramp metering
- Mandatory lane selection
- Speed limit selection
- Increased road size

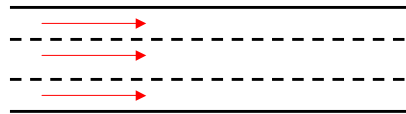


Car-centered traffic management

- Cars take actions to improve traffic
- Lane change coordination
- Goals:
 - Maintain desired speeds
 - Minimize lane changes
 - Increase throughput

Traffic as a distributed AI task

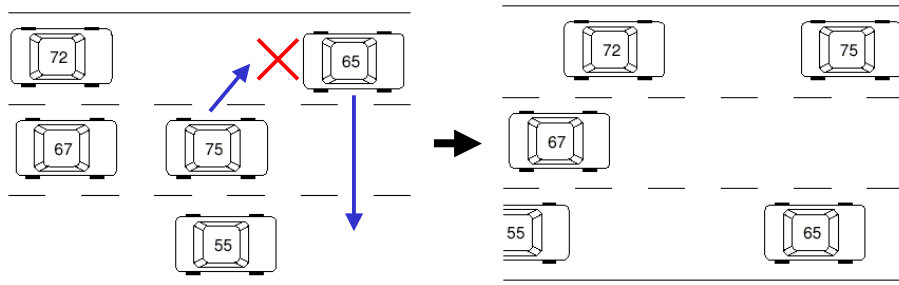
- Car = individual agent 
- Environment = multilane roadway



- Choices = {move left, stay, move right}



An example

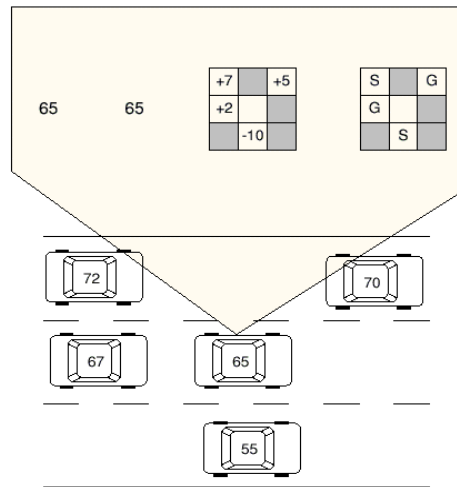




Information available to agents

18 Inputs:

- Current speed
- Desired speed
- 8 Surrounding relative speeds
- 8 smart/greedy



Driver oriented goals

minimize

$$P(C) = \underbrace{\frac{\sum_{t=1}^T \sum_{i=1}^N (\text{actual} - \text{desired})^2}{TN}}_{\text{Maintain desired speed: average difference between desired and actual}} + 60 * \underbrace{\frac{\sum_{i=1}^N L_i}{TN}}_{\text{Minimize lane changes: average \# of lane changes}}$$

actual = actual speed	Li = # lane changes over T steps
desired = desired speed	N = number of cars
T = # time steps	C = set of cars

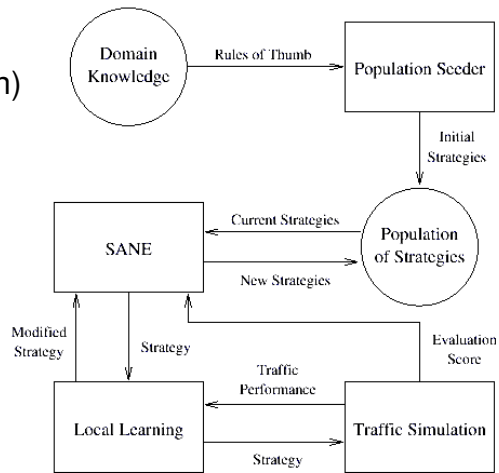
Prefer lane change if speed is wrong by > 8

Learning a control strategy

- SANE (Symbiotic Adaptive Neuro-Evolution)

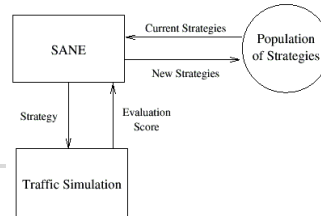
To increase learning efficiency:

- Population seeder
- Local Learning



SANE

- Genetic algorithm
- Learns neural networks
- Evolves the hidden neurons
- These neurons can be combined to form neural networks
- Population members (neurons) are represented by connections with weights

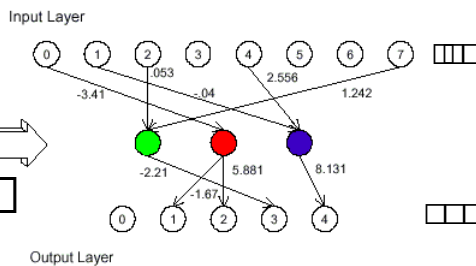


Creating a network from neurons

Three neurons:

label	weight
15	1.242 143 -2.21 2 .053
212	5.811 32 -3.41 151 -1.67
65	-.04 100 2.556 134 8.131

Connection 1 Connection 2 Connection 3



If connection label < 127 then connection goes to an input unit number (label mod I), otherwise to an output unit number (label mod O)

SANE Algorithm

- Break each starting network into component neurons
- Repeat the following for some number of iterations:
 - Calculate average fitness for each neuron (next slide)
 - Rank neurons and pair each neuron in the top quarter with a neuron of higher fitness
 - Create two offspring doing a one point crossover (i.e. swap at one point along length):

parents	15	1.242	143	-2.21	2	.053
	212	5.811	32	-3.41	151	-1.67

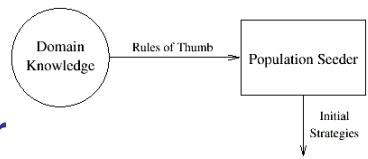
children	15	1.242	143	-3.41	2	.053
	212	5.811	32	2.21	151	-1.67

- Perform random mutation .1% of the time

Calculating average fitness values

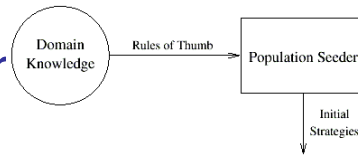
- Set each neuron's fitness to 0
- Repeat some number of times:
 - Select n neurons randomly from the population
 - Create a neural network from selected neurons
 - Evaluate network on given task
 - Add network's score to each neuron's fitness
- Calculate each neuron's average fitness

Population seeder



- Create initial strategies using general domain knowledge
 - Yield right**
 - (desired < 55 mph) & (right open) -> move right
 - (In leftmost) & (desired of car behind > desired) & (right open) -> change right
 - Go around slower cars**
 - (actual of car in front < desired) & (left open) -> change left
 - (actual of car in front < desired) & !(left open) & (right open) -> change right
- Polite strategy = all 4 rules
- Selfish strategy = last 2 rules

Population seeder continued

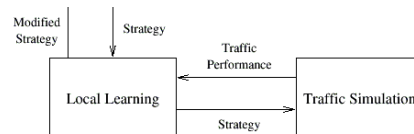


- Create input and output pairs by applying rules of thumb in simulator:



- Randomly initialize some number of networks
- Use backpropagation to train networks based on input output pairs
- Initial strategies: 25% learned and 75% random

Local Learning




- Sample performance every 10 simulated seconds
- If performance difference > some constant (10 in practice), update strategy
 - All choices in 10 second interval are reinforced
 - Similar to population seeder, use backpropagation to reinforce desired behavior
 - If performance difference is negative, reinforce other two choices
 - For example, if move left was chosen, reinforce stay and move right.

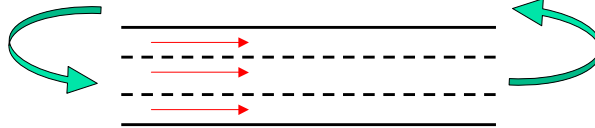
Experiments



- Simulator setup
- Evaluation of intelligent lane selection
 - Varying traffic densities
 - Lane closures
 - 4 lanes
 - Effect of selfish cars
- Contribution of each learning component

The Simulator

- All cars are the same size 
- 13.3 miles of roadway which wraps around



- All cars accelerate and decelerate at the same rate:
 $D = -2.0 \text{ mph/s}$ $A = \frac{10}{\sqrt{\text{speed}}} \text{ mph/s}$ 0-60 in 31 s
60-0 in 30 s
- No curves, hills or ramps
- Cars try and maintain desired speed
- A lane change takes 1 second
- Simulator time step is 1 second

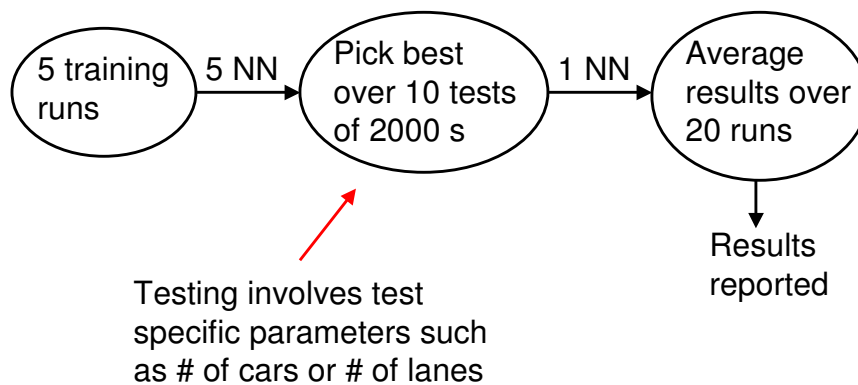


Training

- 400 simulated seconds per evaluation of a neural network
 - 200 randomly dispersed cars
 - Desired speeds selected with mean = 60 mph and standard deviation = 8 mph
 - # smart cars selected randomly with at least 5% smart (all others are selfish)
 - Lane closures last for one mile and are in either rightmost or leftmost lane
- Final Strategy is chosen by keeping each consecutive new “champion” during training and picking the best of these on ten 2000 second evaluations



Testing



Three strategies

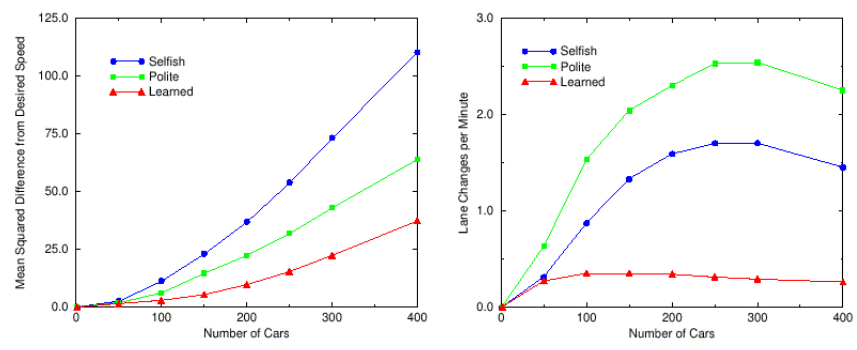


- Learned strategy
- Polite strategy (all 4 rules)
- Selfish strategy (last 2 rules)
 - (desired < 55 mph) & (right open) -> move right
 - (In leftmost) & (desired of car behind > desired) & (right open) -> change right
 - (actual of car in front < desired) & (left open) -> change left
 - (actual of car in front < desired) & !(left open) & (right open) -> change right

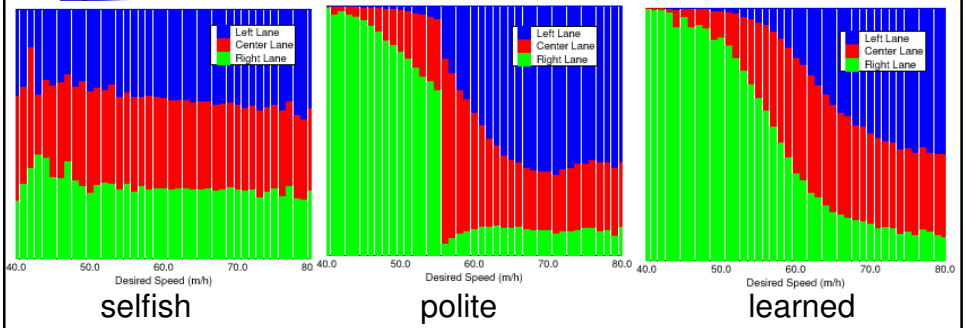
Traffic density

- Tested with 50 to 400 cars on the 13.3 miles of road
 - 50 \approx 1 car every 1.25 miles
 - 400 \approx 10 cars per mile of road
 - Dense \approx 120 cars per mile of road (2 car lengths of separation)
- Remember, trained with 200 cars

Density results



Lane utilization as density increases

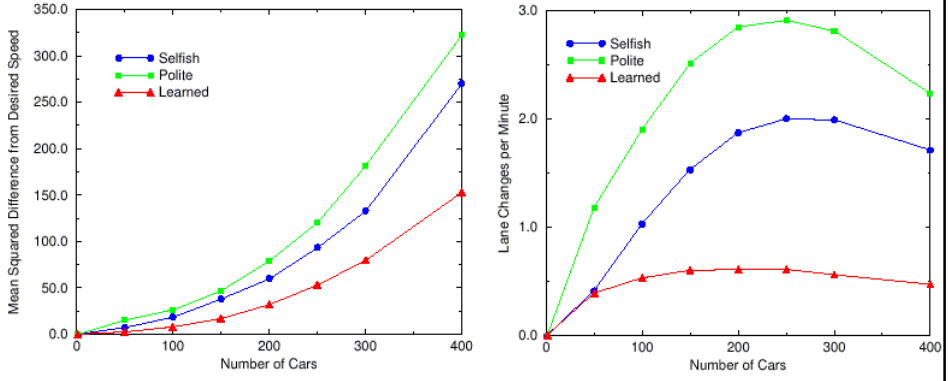


	Left lane	Center lane	Right lane
Overall lane utilization	0.35	0.35	0.30
Selfish	0.35	0.26	0.39
Polite	0.25	0.27	0.48
Learned			

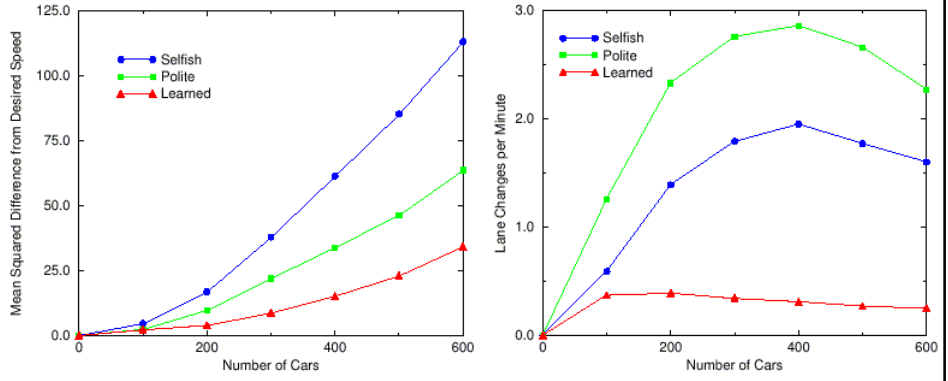
Lane closures



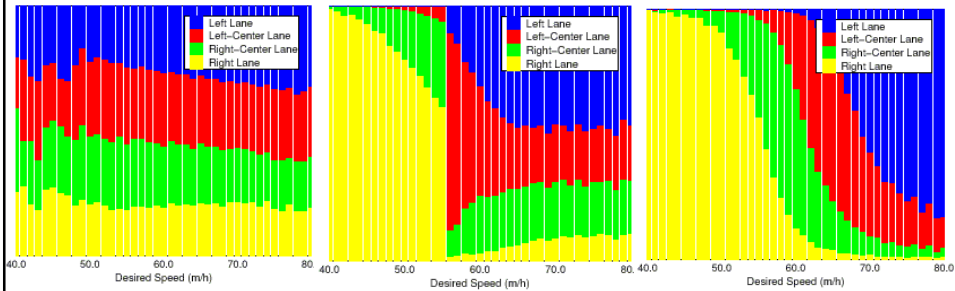
- One mile closed
- Changed every 500 simulated seconds



4 Lane road



Lane utilization with 4 lanes



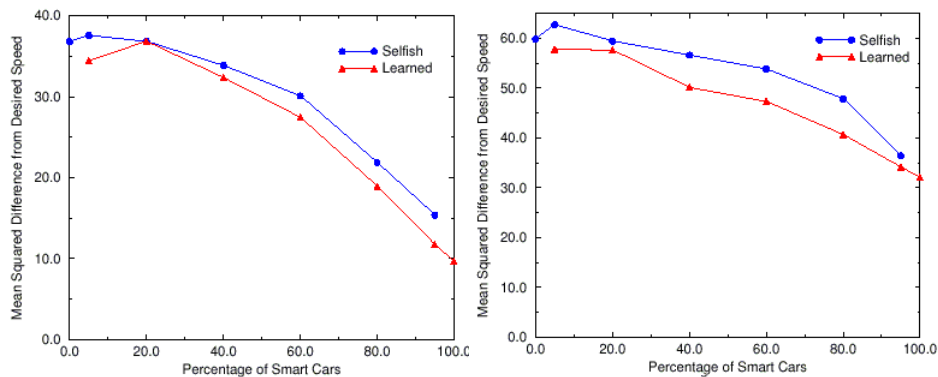
selfish

polite

learned

- Drivers with desired speed of 80 mph drive in left lane 87% of the time vs. 57% for 3 lanes
- Center two lanes are used to organize traffic

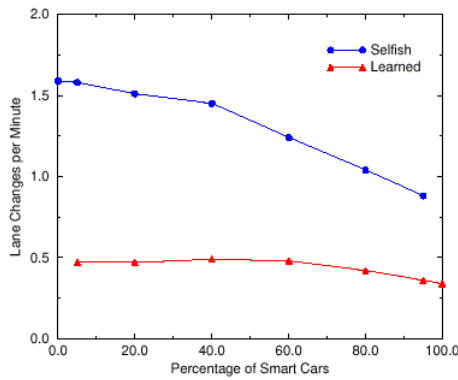
Mixing selfish and learned: speed analysis



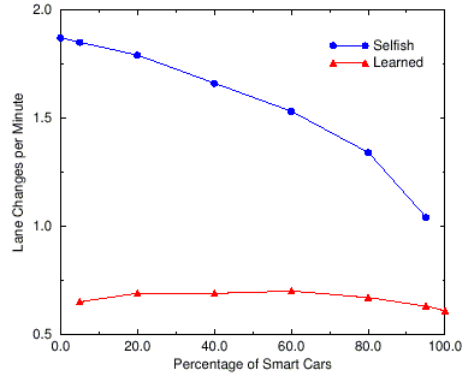
No blocked lanes

Blocked lanes

Mixing



No blocked lanes

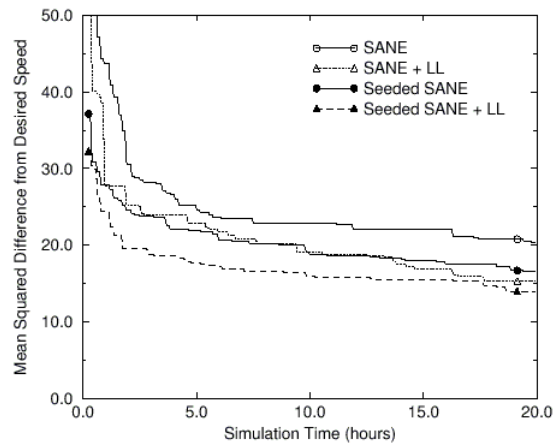


Blocked lanes

Contributions of different learning components

- 3 learning components: population seeder, SANE and local learning module
 - SANE
 - SANE with local learning
 - SANE with population seeder
 - SANE with both local learning and population seeder
- 50 cars on 3.3 mile road without lane blockages
- 20 trial test set: 10 with 100% smart cars and 10 with a random number of smart cars

Different learning components



Future work

- On-ramps and off-ramps
- More realistic driver modeling
 - Constant desired speed
 - Soft threshold for speed tolerance
- Different performance metric (such as throughput)
- Speed control

