Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Improving Hybrid Vehicle Fuel Efficiency Using Inverse Reinforcement Learning

Adam Vogel
Stanford University
av@cs.stanford.edu

Deepak Ramachandran, Rakesh Gupta, Antoine Raux
Honda Research Institute (USA) Inc.
{dramachandran,rgupta,araux}@hra.com

Abstract

Deciding what mix of engine and battery power to use is critical to hybrid vehicles' fuel efficiency. Current solutions consider several factors such as the charge of the battery and how efficient the engine operates at a given speed. Previous research has shown that by taking into account the future power requirements of the vehicle, a more efficient balance of engine vs. battery power can be attained. In this paper, we utilize a probabilistic driving route prediction system, trained using Inverse Reinforcement Learning, to optimize the hybrid control policy. Our approach considers routes that the driver is likely to be taking, computing an optimal mix of engine and battery power. This approach has the potential to increase vehicle power efficiency while not requiring any hardware modification or change in driver behavior. Our method outperforms a standard hybrid control policy, yielding an average of 1.22% fuel savings.

1 Introduction

Hybrid Electric Vehicles (HEVs) reduce fuel usage by using electric storage systems to save part of the energy produced by the engine and regenerative braking. At any time, the proportion of electric and fuel energy can be optimized to improve fuel efficiency. For example, if it is known that the upcoming driving route has stop and go traffic with red lights then there will be opportunities to recharge the batteries from regenerative braking and it is advantageous to use power from the battery. Similarly, if speed is low and the engine efficiency is better at higher RPM, it may be advantageous to run the engine at higher RPM and save the extra energy in the battery.

With an estimated 250 million vehicles in the US, typically driven 12,000 miles/year at an average fuel economy of 20.3 miles/gallon, 150 billion gallons of fuel is annually used in the US (Environmental Protection Agency 2005). If 10% of these vehicles, improved fuel efficiency by 1%, 150 million gallons of fuel worth $500 million could be saved every year.

Current powertrain control solutions in HEVs consider several factors such as the charge of the battery and how efficiently the engine operates at a given speed (Paganelli et al. 2001). These approaches do not take into account future power requirements. Recent research has shown that fuel can be saved if the driving route of the vehicle is known (Brahma, Guezennec, and Rizzoni 2000; Deguchi et al. 2004; Johannesson, Asbogard, and Egardt 2007). However, most of the time the destination of the vehicle is unknown a priori. To solve this problem, we use past driver history to predict the future driving route, and use this data to probabilistically optimize fuel usage. Our system predicts the future path of the vehicle, computes the optimal power split between engine and battery and uses it for a short period of time, repeating the computation periodically.

We are able to predict the path a driver is taking because people often repeatedly drive along the same routes. For example, they commute to work, go to the same grocery stores and take their children to classes regularly every week. Froehlich et al. (Froehlich and Krumm 2008) found that in their in their data set of 250 drivers (restricted to those observed for at least 40 days) nearly 60% of the trips were duplicated in the data. We thus aim to optimize vehicle energy consumption by predicting the route that the driver is likely to take.

For turn prediction, there is related work (Ziebart et al. 2008b; Simmons et al. 2006; Krumm 2008) all of which make strong assumptions. Ziebart (Ziebart et al. 2008b) assumes the destination is given and predicts turns, Simmons (Simmons et al. 2006) uses a database that has 95% of decision points having a single choice, and Krumm (Krumm 2008) requires an intersection to have been visited in the training data in order to make a prediction. In contrast, we learn a model that does not assume that the destination is known, and our model generalizes to unseen data.

We wish to perform energy savings in the background, without modifying user route or behavior. Requiring the user to enter the route they will take ahead of time is unrealistic and increases driver cognitive load. Other past work suggests fuel efficient driving routes to the user. For example, Ganti et al.(2010) identify fuel efficient routes using a car's internal sensors. A notable exception is Kohut et al.(Kohut, Hedrick, and Borrelli 2009), who use predicted traffic information to improve the fuel efficiency of a conventional vehicle.

In the remainder of the paper, we first describe how we learn models of driver behavior using Inverse Reinforcement

Learning. We next define our vehicle simulator which we use to evaluate fuel usage for different control policies. Section 4 describes a dynamic programming algorithm for computing the optimal fuel policy given the driver model and vehicle simulator. Lastly, in Section 5 we present experimental results which show that we successfully learn to predict driver behavior, and that by utilizing this driver model we can reduce overall fuel usage.

2 Driver Modeling using Inverse Reinforcement Learning

In this paper we describe our model of driver behavior learned using the Maximum Entropy Inverse Reinforcement Learning approach. The general MaxEnt IRL algorithm was first presented in (Ziebart et al. 2008a) and an extended application to modeling taxicab driver behavior called PROCAB was described in (Ziebart et al. 2008b). Our method follows the PROCAB system closely.

In PROCAB, the driver's route choice is modeled as a Markov Decision process. States s_j in the MDP correspond to road segments in the network. The actions available at a given state are all the possible turns a_j the driver could make at the intersection at the end of s_j.

Each driver is assumed to have an implicit cost function that expresses his preferences over trip segments and routes. As usual, we represent this function as a linear combination of action features $\boldsymbol{f}_{a_j}$ with a weight vector θ:

$$cost(a_j) = \theta^\intercal \boldsymbol{f}_{a_j}$$

Our features f_{a_j} capture salient aspects of the driving route, where a_j is a transition from road segment s_j to s_{j+1}. These features include

- The identity of the outgoing road segment s_{j+1}. These features model driver preferences for specific roads.
- The type of road s_{j+1} is, such as residential, highway, etc. These model driver preferences like taking the highway versus a service road.
- The angle of turn between s_j and s_{j+1}, discretized into left, straight, right, and U-turn. Using these features we can learn that U-turns are uncommon, and that drivers frequently go straight through an intersection.

We conjoin each of these feature with a time of day feature, which can capture daily routines. For instance, in the morning drivers are likely heading to work and perhaps in the afternoon they drive to pick their child up from school.

Given a route $\varsigma = (s_0, a_0, s_1, a_1, \ldots, a_{n-1}, s_n)$, let $\boldsymbol{f}_\varsigma$ be the sum of the features for each action along the route: $\boldsymbol{f}_\varsigma = \sum_j \boldsymbol{f}_{a_j}$. The driver is assumed to choose a route ς that minimizes the cost function:

$$cost(\boldsymbol{f}_\varsigma) = \theta^\intercal \boldsymbol{f}_\varsigma = \sum_{a_j \in \varsigma} \theta^\intercal \boldsymbol{f}_{a_j}$$

Our problem is to recover the parameters of this function given the demonstrated behavior of the driver in the form of a collection of trajectories ς_i. This problem is known as *Inverse Reinforcement Learning* in the RL literature and has been intensively studied recently (Abbeel and Ng 2004; Ramachandran and Amir 2007). The key challenge in IRL is that it is under-constrained: multiple viable cost functions are possible (in particular uniform zero costs can explain any action). Additional assumptions or constraints are needed to choose among the cost functions. In (Ziebart et al. 2008b), the principle of maximum entropy (Jaynes 1957) is used to identify a distribution over paths given θ that exhibits no additional preferences beyond matching observed behavior:

$$P(\varsigma_i|\theta) = \frac{1}{Z(\theta)} e^{-\theta^\intercal \boldsymbol{f}_{\varsigma_i}} = \frac{1}{Z(\theta)} e^{-\sum_{a_j \in \varsigma_i} \theta^\intercal \boldsymbol{f}_{a_j}}$$

where $Z(\theta)$ is a normalization factor. This yields a stochastic policy where the probability of action a is weighted by the expected exponentiated rewards of all paths that begin with a:

$$P(a|\theta) \propto \sum_{\varsigma : a \in \varsigma_{i=0}} P(\varsigma|\theta)$$

As discussed in (Ziebart et al. 2008a), this approach overcomes the *label bias* problem (Lafferty, McCallum, and Pereira 2001) that affects IRL methods using local action potentials to define the distribution (Neu and Szepesvári 2007; Ramachandran and Amir 2007).

We now maximize the likelihood of the observed data max-entropy distribution defined above w.r.t parameters θ, obtaining:

$$\theta^* = \text{argmax}_\theta L(\theta) = \text{argmax}_\theta \sum_{\text{examples}} \log P(\varsigma|\theta)$$

For our deterministic MDP, this function is convex and its maximum can be found using an online exponentiated gradient ascent algorithm. The gradient of the log likelihood has an elegant representation as the difference between the empirical feature counts and the model's expected feature counts:

$$\nabla L(\theta) = \tilde{\boldsymbol{f}} - \sum_\varsigma P(\varsigma|\theta) \boldsymbol{f}_\varsigma = \tilde{\boldsymbol{f}} - \sum_{a_j} D_{a_j} \boldsymbol{f}_{a_j}$$

where D_{a_j} is the expected number of times we take turning action a_j and $\tilde{\boldsymbol{f}}$ are the empirical feature counts. A straightforward computation of D_{a_j} would require enumerating all paths from a_j. Instead, we use an efficient forward-backward dynamic programming algorithm presented in (Ziebart et al. 2008a) to perform the inference.

3 Driving Simulator

To test our predictive energy optimization strategy, we utilize a standard simulation model of a parallel hybrid electric vehicle. The model comes in three components. The *energy model* calculates how the battery and fuel levels change as a function of power output. The *vehicle dynamics model* calculates the forces acting on the vehicle to calculate the power required to reach a desired acceleration. Lastly, the *driver model* computes the desired acceleration from the driver.

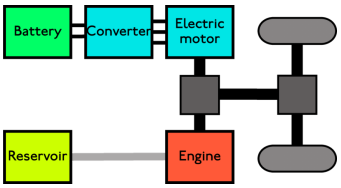

Figure 1: Hybrid vehicle powertrain.

3.1 Energy Model

In a parallel hybrid powertrain (most common at present e.g. Honda Insight), the internal combustion engine and battery are coupled through a differential gear (see Figure 1). The instantaneous power requirement (determined from velocity and acceleration) is supplied by the engine and battery:

$$P_{req}(t) = P_{eng}(t) + P_{batt}(t)$$

$P_{batt} < 0$ corresponds to the case where the engine is charging the battery as well as driving the wheels. The evolution of the battery dynamics can be found by modeling it as a simple circuit (Figure 2). The variation in the State of Charge (SOC) x is proportional to current at the battery terminals:

$$\dot{x}(t) = -\frac{1}{Q_{nom}} I(t)$$

where $I(t)$ is the current (positive during discharge) and Q_{nom} is the nominal charge capacitance, measured in joules. Further,

$$P_{batt}(t) = V_{oc}(x)I(t) - R_o(x)I^2(t)$$

where $V_{oc}(x)$ and $R_o(x)$ are the open-circuit voltage and internal resistance respectively of the battery. Following (Serrao, Onori, and Rizzoni 2011), we set $R_0(x) = 0.01$ ohms, and

$$V_{oc}(x) = 1.4x + 3.18$$

where $x \in [0, 1]$ is the relative charge of the battery and V_{oc} is measured in volts. Combining these equations, the derivative of the SOC is:

$$\dot{x}(t) = -\frac{1}{Q_{nom}} \frac{V_{oc}(x) + \sqrt{V_{oc}^2(x) - 4R_0(x)P_{batt}(t)}}{2R_0(x)} \quad (1)$$

We can now solve for $x(t)$ to update the battery state.

$$x(t+1) = x(t) + \dot{x}(t)$$

We simulate at a rate of 1Hz and use Euler integration to track the SOC over time.

3.2 Vehicle Dynamics Model

Four forces act on the car: the powertrain, rolling friction, air resistance, and gravitational force.

$$F_{car} = F_{eng} - F_{friction} - F_{air} - F_g$$

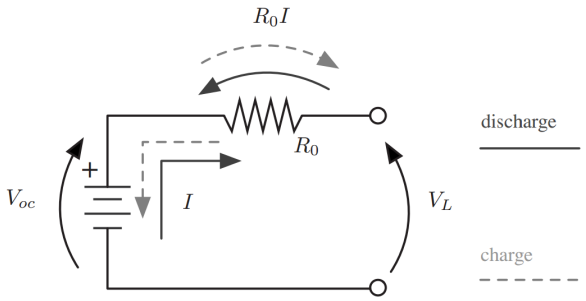

Figure 2: The circuit diagram for the battery.

Constant	Interpretation	Value
m	mass of car	1000 kg
A	surface area of car front	2 m^2
ρ	density of air	1.225 kg/m^3
c_w	drag coeff. of wind resistance	0.42
c_{rr}	coeff. of rolling resistance	0.01

Table 1: The values of physical constants used in our simulation

where

$$F_{car} = ma$$
$$F_{friction} = c_{rr} mg \cos(\theta)$$
$$F_{air} = \frac{1}{2} c_w A \rho v^2$$
$$F_g = mg \sin(\theta)$$

The interpretation and values of the constants used above are shown in Table 1. To compute the power required by the engine, we use the approximation $P = F\bar{v}$ which gives:

$$P_{eng} = ma\bar{v} + c_{rr} mg \cos(\theta)\bar{v} + \frac{1}{2} c_w A \rho \bar{v}^3 + mg \sin(\theta)\bar{v}$$

To model the fuel consumption as a function of required power, we assume that the engine speed is operated at the *Optimal Operating Line* (OOL). This gives a linear mapping from required power to fuel consumption, given by

$$\Delta\text{fuel} = \alpha P_{eng} \quad (2)$$

Where P_{eng} is measured in kW and fuel consumption is measured in gallons per second. We use $\alpha = 0.1$, following (Serrao, Onori, and Rizzoni 2011).

3.3 Driver Model

The last component of our vehicle simulator calculates the desired acceleration of the driver. To generate the driving cycle from the predicted route of the driver we adapt the *Intelligent Driver Model* (IDM) from (Treiber, Hennecke, and Helbing 2000). The IDM computes car acceleration and braking by combining the desired velocity of a driver with the distance to obstacles. Originally developed to model car-following behavior, we utilize the IDM by treating stop-signs and lights as cars with zero velocity. Thus typical behavior is for a vehicle to accelerate from the beginning of a

Figure 3: An overview of our dataset and road network. Colors more towards blue denote high-velocity routes and routes which are more red have lower speed.

road segment up to the speed limit and then decelerate as it approaches the next stopping point.

$$\dot{v} = a(1 - (\frac{v}{v_0})^{\delta}) - b\frac{v^2}{4s^2} \tag{3}$$

where the variables are defined as

- a: maximum acceleration
- v: current velocity
- v_0: desired velocity, assumed to be the speed limit
- b: maximum braking deceleration
- s: distance to next stop
- δ: smoothness parameter, set to 4 in (Treiber, Hennecke, and Helbing 2000).

4 Powertrain Control

Using the probabilistic driver model described in Section 2, we can find an energy policy using a Markov Decision Process for computing an optimal policy combining engine and battery usage in order to minimize energy consumption.

At a given state s_i, we are on a road segment r_i with battery SOC x_i and fuel level f_i. The vehicle simulator reports the instantaneous power requirement P_{req} and the powertrain controller must decide how much power comes from the engine, P_{eng}, and how much from the battery, P_{batt}.

Our action space is $A = \{(P_{eng}, P_{batt}) : P_{eng} + P_{batt} = P_{req}, P_{eng} \geq 0\}$. After we choose (P_{eng}, P_{batt}), the simulator updates the position of the vehicle, state of charge of the battery and fuel level, yielding x_{i+1} and f_{i+1}. Note that r_i, the road segment taken, is chosen by the user and is not under our policy's control; it is instead part of the transition dynamics.

The *reward* of a state $s = (r, x, f)$, which we denote $R(r, x, f)$, is the sum of residual fuel energy and battery energy for a terminal state (when it has reached its destination), and zero for all other states.

Algorithm 1 A sample option Π_{α_1,α_2}. Here α_1 controls the mixture of battery and engine used and α_2 controls the recharging rate of the battery. $P_{charging}$ is the maximum charging power capacity.

Input: Power Required P_{req}, State of charge x_i,
1. **If** $x_i > Capacity$
   (a) $P_{eng} = \alpha_1 \cdot P_{req}$
   (b) $P_{batt} = (1 - \alpha_1) \cdot P_{req}$
2. **else**
   (a) $P_{eng} = P_{req} + \alpha_2 \cdot P_{charging}$
   (b) $P_{batt} = -\alpha_2 \cdot P_{charging}$

Algorithm 2 The Powertrain Controller invoked on state s_i. The current option Π_{α_1,α_2} is a mapping from vehicle state to a powertrain controller action. This option is recomputed every T_Π time steps.

Input: Road segment r_i, position p_i, velocity $\dot{p}_i$, State of charge x_i, Fuel level f_i, Option Π_{α_1,α_2}, integer OptionCntr
1. Compute desired acceleration $\ddot{p}_i$ from IDM$(r_i, p_i, \dot{p}_i)$.
2. Compute desired power P_{req} from $\ddot{p}_i$ using force model.
3. Apply current option $\Pi(P_{req}, x_i)$ to get (P_{eng}, P_{batt}).
4. Update $p_i, \dot{p}_i$ and r_i from $\ddot{p}_i$ using the vehicle kinematics and road network simulator.
5. Update battery charge x_i using Equation (1)
6. Update fuel level using Equation (2)
7. $OptionCntr++$
8. **If** $OptionCntr > T_\Pi$
   (a) Π_{α_1,α_2} = FindBestOption(r_i,p_i, $\dot{p}_i$, x_i,f_i)
   (b) Reset $OptionCntr$ to 0.

Observe that we have not included a cost term for work done in moving to the destination. Destinations closer to the origin will have higher value than those further away. However, since our powertrain controller cannot affect the choice of destination or the vehicle motion in any way, this cost will be "marginalized out" of the expected value function comparison. Thus we do not account for it in our objective function.

We use our driver model to predict future road segments r_{i+1}, which impact the future required power. We compute the probability of the next road segment r_{i+1} by

$$P(r_{i+1}|r_i) \propto \exp(\theta^\intercal \boldsymbol{f}_{(r_i, r_{i+1})})$$

where θ are the weights learned by our IRL algorithm.

Our goal is to maximize the *value function*, the expected sum of rewards for all future states. By Bellman's equations, this is given by:

$$V(r_i, x_i, f_i) = R(r_i, x_i, f_i) + \sum_{r_i} P(r_{i+1}|r_i) V(r_{i+1}, x_{i+1}, f_{i+1}) \tag{4}$$

Computation of the value function by dynamic programming is inefficient as this would involve updates for all possible states. Since we only care about states reachable from the current initial state, we use a forward search algorithm with a receding time horizon T, to find an approximately optimal control for the current state.

To find a useful powertrain policy, T must be rather large (comparable to the length of the trip) which means a naive search would need large search depth. Instead we use *options* (Algorithm 1), which are temporally extended actions, to reduce the complexity of the search space. We consider options of the form Π_{α_1,α_2}, where α_1 is the proportion of P_{req} supplied by the engine and α_2 controls the rate of recharging of the batteries. This treats the fuel optimization problem as a *semi-Markov Decision Process* (Sutton, Precup, and Singh 1998). The controller (Algorithm 2) applies the selected option over multiple time steps, updating it by forward search (Step 8a, Equation (4)) every $T_\Pi = 20$ seconds.

5 Experiments

In this section we describe a GPS driving dataset we collected. We use this dataset to evaluate our turn prediction model and the powertrain control policy we derive from it.

5.1 Dataset

To build models predicting driver behavior, we had 12 drivers carry a GPS logger in their car for four weeks in the San Francisco Bay Area, yielding approximately 380,000 GPS readings over 9,000 kilometers of driving. Figure 4 shows some overall statistics for each driver. Drivers took an average of 144.83 trips, visiting 31.25 unique destinations. Using an 80% training / 20% testing split, an average of 38.66% of the destinations in the testing data were not observed in the training data. Figure 3 shows a geographic overview of our dataset. We utilize the Open Street Map database (OSM 2012) for road network data, a freely available road map generated by volunteers.

We first segmented each driver's data into trips, judging a trip to end if the vehicle does not move for 5 minutes. This value was set to avoid false positives at stop lights and in stop-and-go traffic while capturing most actual trip destinations.

Once we have the GPS data segmented into trips, we then associate each GPS reading with a road segment using an HMM, modeled after (Letchner, Krumm, and Horvitz 2006). In this HMM, the observations are GPS readings o_i and the hidden variables are locations on road segments s_i. We associate each GPS reading o_i with a location on a road segment s_i, where $P(o_i|s_i)$ is a normal distribution over the Euclidean distance from o_i to s_i, with mean zero and a standard deviation of 15 meters. The transition probability $P(s_i|s_{i-1})$ between road segments is proportional to the straight-line distance from s_{i-1} to s_i divided by the on-road distance between s_{i-1} and s_i. This gives lower probability to road locations which are close in space but not close by the road network, which discourages the model from selecting circuitous routes.

5.2 Driver Model

We first present results for how accurate our driver model is. We evaluate our model for each driver independently, using the first 80% of the data for training and the remaining 20% for testing.

The main intrinsic evaluation metric we are interested in is turn accuracy. Our powertrain control strategy is most dependent on being able to accurately predict the near future. We compare our IRL driver model with a random baseline, where the driver takes a random turn at each intersection.

Figure 5 shows turn prediction accuracy for different drivers in our data. Our model gets an average turn prediction accuracy of 0.65, which is better than the baseline, which gets 0.39 accuracy. Secondly, we evaluate our driver prediction model for the next 5 turns. At each road segment in a driver's route, we use our two models to predict the next 5 turns. We use a strict evaluation, where an example is judged correct only if we get the next 5 turns exactly right. Here the IRL model significantly outperforms the baseline.

Our turn prediction accuracy is below results reported by (Ziebart et al. 2008b; Simmons et al. 2006; Krumm 2008) all of which make strong assumptions to get turn prediction accuracy above 90%. In (Ziebart et al. 2008b) the destination is already known, and a single model is learned from all driver data which makes the turn prediction significantly simpler. In (Simmons et al. 2006), 95% of the data has a single option and only the remaining 5% has multiple options for the driver to turn (i.e. actual intersections). (Krumm 2008) train their Markov model by computing $p(x_i|x_{i-1})$ using previous trips that went through the same intersection. Their model would not work on places not seen in the training data. They also do leave-one-out testing, which means they most likely have seen the intersection in the training data.

5.3 Powertrain Controller

To evaluate our powertrain controller, we used the simulator described in Section 3 on the driving data we collected. Our overall evaluation metric is the amount of energy consumed for a driver's routes. We do this by measuring the amount of energy in the fuel used by the driver, minus the energy remaining in the battery at the end of the trial.

We compare our predictive policy with a baseline policy, the Charge-Depleting Charge-Sustaining (CDCS) policy (Sciarretta and Guzzella 2007). Above a given threshold SOC, the CDCS policy operates in *charge-depleting mode*, where it uses the battery as much as possible. After the battery charge drops below the threshold, the CDCS policy switches to *charge-sustaining mode*, where it only uses the battery to meet power requirements that the engine alone cannot handle. By tuning on the training data, a charge threshold of 15% was found to be optimal for energy efficiency, though it should noted that most production car models set a higher threshold to maximize battery life (Markel 2011).

We used the option-based powertrain controller (Algorithm 1) with parametrized options as described in Algorithm 2. The optimization searches for the best sequence of options chosen from a set of 3-5 options over a horizon time of $T = 30$ minutes. The duration of each option was 20 seconds.

Figure 5 shows the energy used for the baseline policy compared with the predictive powertrain policy. Our method results in savings for all drivers, with the amount of savings

Driver	Trips	Distance (km)	Unique Destinations	% Unseen Test Destinations
1	105	6649	35	64%
2	197	10213	35	29%
3	214	1026	27	22%
4	77	7764	27	50%
5	247	6949	51	42%
6	89	17275	30	33%
7	126	7139	41	38%
8	114	9189	17	60%
9	222	10249	42	33%
10	176	8180	26	38%
11	41	3605	14	33%
12	130	2486	30	22%
Mean	**144.8**	**7560.3**	**31.3**	**38.7**

Figure 4: Dataset statistics for our GPS dataset. Although drivers frequently repeat driving routes, an average of 38.55% of destinations in the test set are not seen in the training set. We consider two destinations to be the same if they are within 200m of each other. This threshold accounts for both GPS sensor noise and the fact that drivers often park in different spots for the same destination.

Driver	IRL Turn Accuracy	Baseline Turn Accuracy	IRL Turn Accuracy @ 5	Baseline Turn Accuracy @5	Energy Used (Policy)	Energy Used (Baseline)	Energy Savings (%age)
1	0.73	0.38	0.31	0.02	2797	2822	0.89
2	0.70	0.39	0.32	0.02	5609	5692	1.47
3	0.32	0.25	0.06	0.02	10023	10036	0.13
4	0.81	0.42	0.49	0.02	6479	6631	2.35
5	0.80	0.43	0.39	0.02	417	424	1.68
6	0.68	0.39	0.34	0.02	1655	1671	0.97
7	0.73	0.41	0.29	0.01	6932	7065	1.92
8	0.63	0.42	0.25	0.02	8407	8500	1.11
9	0.45	0.41	0.06	0.02	12891	12944	0.41
10	0.74	0.40	0.34	0.02	753	762	1.20
11	0.80	0.40	0.35	0.02	7240	7407	2.31
12	0.39	0.34	0.03	0.01	4808	4817	0.19
Mean	0.65	0.39	0.27	0.02	5667	5730	**1.22**%

Figure 5: This table shows the performance of our driver model and the % of energy saved by using our powertrain controller compared to the baseline Charge-Depleting Charge-Sustaining (CDCS) policy. The IRL driver model for turn prediction outperforms the baseline. Energy is measured in megajoules. The quality of the driver model is critical to the fuel savings. We find a strong correlation between turn prediction accuracy and fuel savings, with a Spearman rank-order correlation coefficient of .8819 (p-value 1.48e-4).

being dependent on the particular routes they took and the accuracy of the turn prediction. Averaged across all drivers, our method saves 1.22% on overall energy usage. Though relatively small, these energy savings present the advantage of not requiring any change in behavior from the driver and no hardware modifications to the drive train. A Wilcoxon signed-rank test shows that our method outperforms the baseline with p-value 0.0031. Furthermore, we find a strong correlation between turn prediction accuracy and fuel savings, with a Spearman rank-order correlation coefficient of 0.8819 (p-value 1.48e-4).

6 Conclusion

In this paper we demonstrated that we can improve the energy efficiency of hybrid electric vehicles by utilizing predictive models of future driving requirements. Although the fuel savings from our model are modest, our method does not require any change in user behavior. Given the environmental and economic costs associated with fossil fuel power, any possible energy saving strategy is worth exploring.

The main challenge going forward is testing our approach in a real-world vehicle. Implementing our powertrain policy in a vehicle computer would required dealing with strict memory constraints and realtime computing demands. Instead of explicitly exploring the future state space, we could perhaps generate simpler policies which approximate our powertrain controller.

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04. New York, NY, USA: ACM.

Brahma, A.; Guezennec, Y.; and Rizzoni, G. 2000. Optimal energy management in series hybrid electric vehicles. *Proceedings of the American Control Conference* 1:60–64.

Deguchi, Y.; Kuroda, K.; Shouji, M.; and Kawabe, T. 2004. HEV charge/discharge control system based on navigation information. *International Congress and Exposition On Transportation Electronics* 1.

Environmental Protection Agency. 2005. Epa emission facts: Calculating emissions of greenhouse gases: Key facts and figures. http://nepis.epa.gov/Adobe/PDF/P1001YTV.PDF.

Froehlich, J., and Krumm, J. 2008. Route prediction from trip observations. In *Proceedings of SAE World Congress*.

Ganti, R. K.; Pham, N.; Ahmadi, H.; Nangia, S.; and Abdelzaher, T. F. 2010. Green GPS : A participatory sensing fuel-efficient maps application. In *Proceedings of the 8th international conference on Mobile systems, applications, and services (MobiSys)*.

Jaynes, E. T. 1957. Information theory and statistical mechanics. *Phys. Rev.* 106(4):620–630.

Johannesson, L.; Asbogard, M.; and Egardt, B. 2007. Assessing the potential of predictive control for hybrid vehicle powertrains using stochastic dynamic programming. *IEEE Transactions on Intelligent Transportation Systems* 8(1):71–83.

Kohut, N.; Hedrick, K.; and Borrelli, F. 2009. Integrating traffic data and model predictive control to improve fuel economy. In *12th IFAC Symposium on Control in Transportation Systems*.

Krumm, J. 2008. A markov model for driver turn prediction. In *Proceedings of SAE World Congress*, volume 2193.

Lafferty, J.; McCallum, A.; and Pereira, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*.

Letchner, J.; Krumm, J.; and Horvitz, E. 2006. Trip router with individualized preferences (TRIP): Incorporating personalization into route planning. In *AAAI*. AAAI Press.

Markel, T. 2011. *Plug-In Hev Vehicle Design Options and Expectations: Zev Technology Symposium, California Air Resources Board, Sacramento, CA, September 27*.

Neu, G., and Szepesvári, C. 2007. Apprenticeship learning using inverse reinforcement learning and gradient methods. In Parr, R., and van der Gaag, L. C., eds., *UAI*, 295–302. AUAI Press.

OSM. 2012. Open street map project. http://www.openstreetmap.org/.

Paganelli, G.; Ercole, G.; Brahma, A.; Guezennec, Y.; and Rizzoni, G. 2001. General supervisory control policy for the energy optimization of charge-sustaining hybrid electric vehicles. *JSAE Review* 22(4):511–518.

Ramachandran, D., and Amir, E. 2007. Bayesian inverse reinforcement learning. In *Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*.

Sciarretta, A., and Guzzella, L. 2007. Control of hybrid electric vehicles. *Control Systems, IEEE* 60–70.

Serrao, L.; Onori, S.; and Rizzoni, G. 2011. A comparative analysis of energy management strategies for hybrid electric vehicles. *Journal of Dynamic Systems, Measurement, and Control* 133(3):031012.

Simmons, R.; Browning, B.; Zhang, Y.; and Sadekar, V. 2006. Learning to predict driver route and destination intent. In *Proceedings of IEEE Intelligent Transportation Systems Conference (ITSC)*.

Sutton, R. S.; Precup, D.; and Singh, S. 1998. Between mdps and semi-mdps: Learning, planning, and representing knowledge at multiple temporal scales.

Treiber, M.; Hennecke, A.; and Helbing, D. 2000. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E* 62:1805.

Ziebart, B. D.; Maas, A.; Bagnell, J. A.; and Dey, A. K. 2008a. Maximum entropy inverse reinforcement learning. In *Proceedings of AAAI Conference*.

Ziebart, B. D.; Maas, A.; Bagnell, J. A.; and Dey, A. K. 2008b. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *Proc. Ubicomp*, 322–331.