Optimizing a Start-Stop Controller Using Policy Search

Noel Hollingsworth[1] **Jason Meyer**[2] **Ryan McGee**[2] **Jeffrey Doering**[2]
George Konidaris[1] and **Leslie Kaelbling**[1]

Computer Science and Artificial Intelligence Laboratory[1]
Massachusetts Institute of Technology
{nholling, gdk, lpk}@mit.edu

Vehicle Controls, Research & Advanced Engineering[2]
Ford Motor Company
{jmeyer84, rmcgee3, jdoering}@ford.com

Abstract

We applied a policy search algorithm to the problem of optimizing a start-stop controller—a controller used in a car to turn off the vehicle's engine, and thus save energy, when the vehicle comes to a temporary halt. We were able to improve the existing policy by approximately 12% using real driver trace data. We also experimented with using multiple policies, and found that doing so could lead to a further 8% improvement if we could determine which policy to apply at each stop. The driver's behaviors before stopping were found to be uncorrelated with the policy that performed best; however, further experimentation showed that the driver's behavior *during* the stop may be more useful, suggesting a useful direction for adding complexity to the underlying start-stop policy.

Introduction

Energy consumption is one of the most important problems facing the world today. Most current energy sources pollute and are not renewable, which makes minimizing their use critical. The transportation industry accounts for a significant fraction of the non-renewable energy consumed within the US. Therefore, any improvements to the efficiency of cars being driven could have a major effect on total energy consumption.

This reduction can be significant even if the gain in efficiency is small. In 2012 Ford sold over 5.6 million automobiles (Ford Motor Company 2013), with an average fuel efficiency of 27.1 miles per gallon. Assuming these vehicles are driven 15,000 miles a year, a .01 miles per gallon improvement in efficiency would saves more than 980,000 gallons of gasoline per year.

One approach that automobile manufacturers have used to improve fuel efficiency is to create start-stop systems for use in their cars. A start stop system turns off the vehicles engine when the driver applies the brakes and stops the vehicle and then restarts the engine when the driver lifts their foot off the brake pedal. By temporarily shutting down the engine, these systems avoid wasting fuel to idle the engine. It is estimated that more than half of the light duty vehicles sold by 2022 will be equipped with start-stop systems (Navigant Research 2013). Although the systems are generally efficient, shutting the car's engine down for very short periods of time can result in a net energy loss due to the additional energy required to restart the engine. By not turning off the car's engine as often for shorter stops and turning it off more often for longer stops, it may be possible to conserve more energy. Current start-stop systems are capable of fuel economy improvements of about 3.5% (Wang, McGee, and Kuang 2013).

We applied a policy search algorithm to optimize Ford's existing start-stop controller by approximately 12% using real driver trace data. We also experimented with using multiple policies, and found that doing so could lead to a further 8% improvement if we could determine which policy to apply at each stop. The driver's behaviors before stopping were found to be uncorrelated with the policy that performed best; however, further experimentation showed that the driver's behavior *during* the stop may be more useful, suggesting a useful direction for adding complexity to the underlying start-stop policy.

Start-Stop Controllers

Ford uses a rule-based start-stop controller, similar to the one shown in Figure 1. We are concerned here with start-stop controllers for automatic transmission systems, where only brake and accelerator pedal inputs are available (Wang, McGee, and Kuang 2013). For manual transmission systems, the clutch and shift lever inputs are the primary inputs to shut off or restart the engine. This controller is parametrized by the variables used in the control rules; our aim is to find settings of those parameters that minimize cost.

More formally, the goal is to pick a parameter vector θ for the policy that minimizes $\mathbb{E}[C|\theta]$ over all drivers in the population, where C is composed of three factors:

$$C = c_1 + c_2 + c_3. \quad (1)$$

The first factor, c_1, represents the amount of time the engine is kept running when the car is stopped, and is equal to the time that car has spent stopped minus the time the combustion engine was turned off during a driving session. The second factor, c_2, represents the cost of restarting the car's engine, and is equal to twice the number of times the car's engine has been restarted during a period of driving. This term ensures that for stopping durations less than

```
if CarStopped then
    if EngineOn and BrakePressure > θ0 then
        TURN ENGINE OFF;
    if EngineOn and BrakePressure > θ1 and
    AverageSpeed < θ2 then
        TURN ENGINE OFF;
    if EngineOff and BrakePressure < θ3 then
        TURN ENGINE ON;
    if EngineOn and Δ BrakePressure < θ4 then
        TURN ENGINE OFF;
```

Figure 1: An example rule-based start-stop controller.

two seconds it is desirable (i.e., lower cost) to keep the engine running. The last factor, c_3, penalizes situations where the engine is not restarted fast enough to meet the drivers torque demand. In practice, the engine must be restarted before the driver begins to apply the accelerator pedal. Therefore, each time the engine is restarted, c_3 is incremented by $10 \times \max(0, RestartAnticipationTime - 0.15)$ where $RestartAnticipationTime$ is the amount of time before the accelerator pedal is applied that an engine restart is commanded.

Given a parameter vector θ, this controller is run 100 times a second during a stop in order to determine how to proceed. The controller is responsible for both turning the engine off when the car comes to a stop, as well as turning the engine back on when the user is about to accelerate. This helps to minimize c_3 in the cost function, but may also lead to more complicated behavior, as the controller may turn the engine off and on multiple times during a stop if it believes the car is about to accelerate but the car does not.

The inputs to the system are three data traces: vehicle speed, the accelerator pedal position, and brake pressure, which are each sampled at 100 Hz. Figure 2 shows a graph of an example of input data.

We were given access to data obtained by Ford during its participation in EuroFOT (the European Large-Scale Field Operational Test on In-Vehicle Systems) (Csepinszky 2011). The data consisted of a collection of traces for 19 different drivers, each with on average 100 hours of driving data obtained under real-world conditions, in automatic transmission vehicles without start-stop controllers. The presence or absence of a start-stop system should not affect the behavior of the driver and intrusions of the system are penalized by the cost function. Therefore, the use of data from non start-stop vehicles should not significantly impact the results.

Reinforcement Learning and Policy Search

Reinforcement learning (Sutton and Barto 1998) is a machine learning paradigm concerned with optimizing the performance of agents that are interacting with their environment and attempting to maximize an accumulated reward or minimizing an accumulated cost. Reinforcement learning problems are typically formulated as a Markov decision process (MDP) described by a tuple:

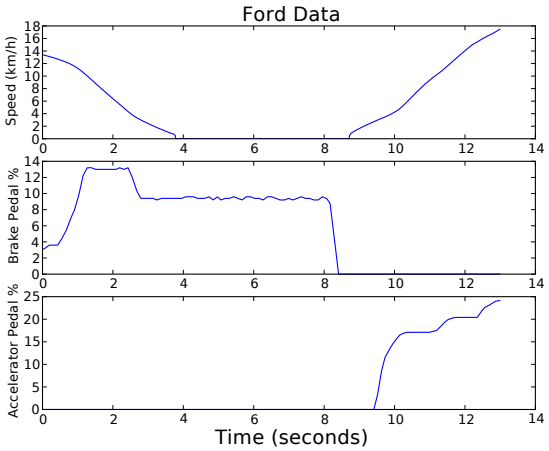


Figure 2: A slice of vehicle data, showing a driver coming to a stop and then accelerating out of it.

$$\langle S, A, P, R \rangle, \tag{2}$$

where S corresponds to the set of states that the agent may be in; A is the set of all possible actions the agent may take; P is the probability distribution for each state-action pair, with $p(s'|s, a)$ specifying the probability of entering state s', having been in state s and executed action a; and R is the reward function $R(s, a)$ specifying the reward received by the agent for executing action a in state s. The goal is to find a policy π mapping states to actions that leads to the highest long-term reward, defined as the discounted sum of the rewards received at all future steps:

$$R^{\pi} = \sum_{t=0}^{\infty} \gamma^t r_t, \tag{3}$$

where r_t denotes the reward gained at time step t, and $\gamma \in (0, 1]$ is a discount factor. In the reinforcement learning setting the agent operates without an explicit model of its environment, so it must improve its policy by directly interacting with the environment.

The most common family of reinforcement learning methods are based on learning a *value function* expressing the agent's predicated reward from a given state (or state-action pair). A value function learned for a given policy can be used to improve that policy, in a process called *policy iteration* (Sutton and Barto 1998). Unfortunately, value function methods are difficult to use directly when the policy to be optimized is of a given parametrized form. In such cases, we can use policy search methods.

Policy Search Methods

Policy search methods optimize a predefined policy $\pi(\theta)$ with given a fixed vector θ of open parameters. As before, we maximize return, or equivalently minimize a cost function $C(\theta)$. Instead of choosing actions directly, the goal is to determine a parameter vector for the policy, which then

chooses actions based on the state of the system. This allows for a much more compact policy representation, because the number of elements of θ may be much smaller than those required to learn a corresponding value function. Additionally, background knowledge can be included in the policy, for example restricting the set of reachable policies to a class known to be sensible for the application in question.

Policy Gradient Algorithms. Policy gradient algorithms are the most popular subset of policy search algorithms. Here, cost is minimized by estimating the derivative of the reward function with respect to the parameter vector, and then descending the cost surface repeatedly until the algorithm converges to a local minimum. If the derivative can be computed analytically and the problem is an MDP then the following equation, known as the *policy gradient theorem* (Sutton et al. 2000), holds:

$$\frac{\partial R}{\partial \theta} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a). \quad (4)$$

Here, $d^\pi(s)$ is the summed probability of reaching a state s using the policy π, with some discount factor applied, which can be expressed as the following, where s_0 is the initial state, and γ is the discount factor:

$$d^\pi(s) = \sum_{t=0}^{\infty} \gamma^t Pr\{s_t = s | s_0, \pi\}. \quad (5)$$

Most algorithms for policy search rely on the policy gradient theorem. Unfortunately, the stop-start controllers we used are not differentiable; moreover, the actual policy available to us was an executable black-box, which only allowed us to execute the policy with a given set of parameters. We therefore used the derivative-free policy search algorithm due to by Kohl and Stone (2004). In place of an analytical gradient, it obtains an approximation of the gradient by sampling, an approach first used in the policy search context by Williams (1992).

The algorithm, given in Figure 3, starts with an arbitrary initial parameter vector θ. It then repeatedly creates random perturbations of the parameters and determines the cost of using those policies. A gradient is then interpolated from the cost of the random perturbations. Next, the algorithm adds a vector of length η in the direction of the negative gradient to the policy. The process repeats until the parameter vector converges to a local minima, at which point the algorithm halts.

The algorithm has three parameters that must be tuned. The first of these is the learning rate η. If the learning rate is too low the algorithm converges very slowly, while if it is too high the algorithm may diverge, or it may skip over a better local minimum and go to a minimum that has higher cost. The second parameter is the amount by which each parameter is randomly perturbed. If the value is set too low the difference in costs due to noise may be much greater than the differences in cost due to the perturbation. If it is set too high the algorithm's approximation of a gradient may no longer be accurate. The final parameter that must be set is the initial policy vector. The policy gradient algorithm converges to

$\theta \leftarrow initialParameters$;
while *not finished* **do**
  create $R_1, ..., R_m$ where $R_i = \theta$ with its values randomly perturbed;
  determine$(C(R_1), C(R_2), ..., C(R_m))$;
  for $i \leftarrow 1$ **to** $|\theta|$ **do**
    `// Perturb θ, retain costs.`
    AvgPos $\leftarrow$ average(C) $\forall R$ where $R[i] > \theta[i]$;
    AvgNone $\leftarrow$ average(C) $\forall R$ where $R[i] = \theta[i]$;
    AvgNeg $\leftarrow$ average$(C\ )\forall R$ where $R[i] < \theta[i]$;
    `// Determine` $\partial C/\partial\theta_i$
    if AvgNone $>$ AvgPos *and*
    AvgNone $>$ AvgNeg **then**
      $change_n \leftarrow 0$;
    else
      $change_n \leftarrow$ AvgPos - AvgNeg;
  `// Ascend differential.`
  change $\leftarrow \frac{\text{change}}{|\text{change}|} * \eta$;
  $\theta \leftarrow \theta +$ change;

Figure 3: Kohl and Stone's policy search algorithm.

a local minimum, and the starting policy vector determines which local minimum the algorithm will converge to. If the problem is convex there will only be one minimum, so the choice of starting policy will not matter, but in most practical cases the problem will be non-convex. One solution to this problem is to restart the search from many different locations, and choose the best minimum that is found. When the policy is found offline this adds computation time, but is otherwise relatively simple to implement. We adjusted all three parameters by hand to maximize performance.

Results

Since the EuroFOT data consisted of data from several drivers, we held out data to test against both performance on drivers with data in the training set, and also on drivers that are not present in the training set. We therefore used a training set of 80% of the EuroFOT data from 15 randomly selected drivers, out of 19 in the data. We ran policy search 35 times, 5 times starting with Ford's existing policy and 30 times starting with random policies. We ran the algorithm for 100 iterations and recorded the policy with the lowest cost on the test set (the remaining 20% of driving sessions from drivers in the training set, and all data from the 4 drivers completely removed from the training set). The resulting learning curve for a single policy, averaged over 35 trials, is shown in Figure 4, and the results obtained on the test sets are shown in Table 1. These results show that we can expect an approximate reduction of 12% in cost by replacing Ford's existing start-stop controller calibration with the one found using policy search.

For a randomly selected validation driver, the numerically optimized calibration policy typically reduced the amount of time the engine was left idling with the vehicle stopped

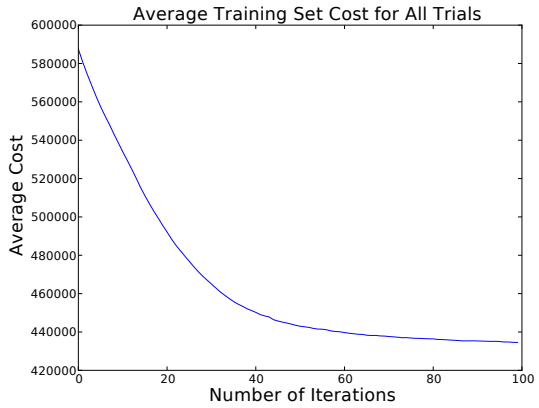


Figure 4: Cost, averaged over 35 trials, as policy search proceeds over 100 iterations.

Data Set	Original Cost	Optimized	Improved
Training Set	494397	424515	14.1%
Different Drivers	90329	77895	13.8%
Same Drivers	121205	105899	12.6%

Table 1: A comparison of the performance of Ford's existing start-stop controller calibration and that obtained using policy search. Comparisons are shown against held-out data from the same set of drivers used during optimization (Same Drivers), and on drivers whose data was completely withheld from training (Different Drivers).

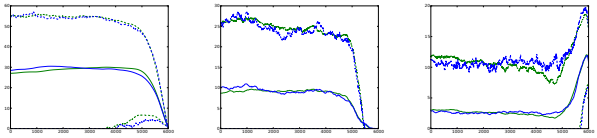
by between 2%–10%, depending on the drive cycle. Additionally, the policy also improved the engine shutdown and restart consistency and reduced the number of exceptionally early or late shutdown and restart events by as much as 20%.

Experiments with Multiple Policies

So far we have considered optimizing a single policy across all drivers. As a first step in improving this, we considered using multiple policies. Our intuition was that there might be multiple "driver modes" in the data (for example, one for highway driving, and one for urban driving) that we might be able to use to improve performance. We experimented with an algorithm based on expectation-maximization (Bishop 2006) to identify driver modes. This algorithm generated two random start policies, then repeatedly found the per-stop cost for each policy and applied policy search to both policies, including only the stops where each policy was best. We repeated this process 35 times, and obtained the best pair of policies.

We found two policies, one of which (policy 1) was assigned to 80% of the stops, and another (policy 2) which was assigned to 20% of the stops. A perfect assignment of these policies to the stops in the data would result in a further 8% reduction in cost. We tried predicting the right policy to apply to a stop based on a set of 60 features provided by Ford that included information about both the previous stop and driving behavior immediately before the target stop.

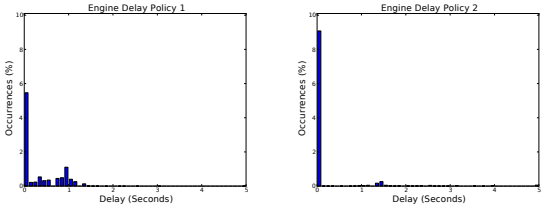
Unfortunately, none of these features had any correlation with the policy to choose next, so we were unable to do better than always predicating the majority policy (policy 1). To understand whether the problem was because we did not have the right features, or because the drivers behavior before the stop just did not predict the right policy to use, we plotted the mean, 90th percentile, and 10th percentile speed, accelerator pressure, and brake pressure values for the 60 seconds before the car came to a stop. These are shown in Figure 5.



(a) (b) (c)

Figure 5: Distributions for speed (a), accelerator pressure (b) and brake pressure (c) in the 60 seconds before a car comes to a stop. The solid line shows the mean value, while the dashed lines show the 90th and 10th percentiles (sometimes not visible as they are at zero). Blue lines are for stops assigned to policy 1, and green lines are for for stops assigned to policy 2. The x-axis is in hundredths of a second.

As these graphs show, the gross statistics of the drivers behavior do not really differ in the full minute before a stop, and we therefore cannot expect to be able to predict the right policy to choose based on that behavior. We also added a fictitious feature corresponding to the actual length of the stop to classification, in order to determine whether or not the policies were specialized to the length of the stop. This feature was also found to be uncorrelated to the classification, from which we concluded that the difference between the two policies has to do with their engine restart behavior. We therefore plotted a histogram of the delay between the driver pressing the accelerator pedal and engine restarting for policy 1 and policy 2, shown in Figure 6.



(a) (b)

Figure 6: Histograms for the delay between the driver pressing the accelerator pedal and the engine restarting, for stops allocated to policy 1 (a) or policy 2 (b).

These two graphs show that both controllers generally result in a very low delay when they are the right controllers to use. However, when we use policy 2 (the minority policy) for stops when policy 1 was the preferred policy, we see a very different histogram of delay lengths in Figure 7.

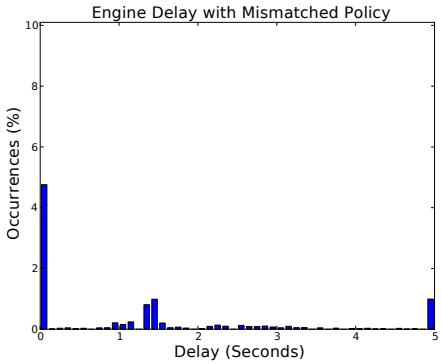


Figure 7: A histogram if the delay between the driver pressing the accelerator pedal and the engine restarting, for stops allocated to policy 1 but executed using policy 2.

Using policy 2 at the wrong time can lead to a long delay in the engine restarting. Note that in this case the value 5 is a catch-all that includes delays longer than 5 seconds, and delays where the driver wanted the car started but took their foot off the pedal before that was completed. Policy 2 is less sensitive to the driver slowly releasing the brake pedal (i.e. slow rate of brake pressure decrease). This means that it does not restart the engine unnecessarily early in 20% of cases. However, in 80% of the cases, when the brake pedal is released at a comparatively slow rate, policy 2 tends to do a poor job of anticipating the driver fully releasing the brake pedal and thus restarts the engine too late. These results suggest that a good way to improve performance is to add rules to the start-stop controller policy to make it adaptive to the patterns of pressure that the driver may exert on the accelerator pedal during the stop.

Related Work

Artificial intelligence techniques have seen extensive use in automative applications (Prokhorov 2008), for example in predicting engine power and torque for automated tuning (Vong, Wong, and Li 2006), learning to diagnose faults (Murphey et al. 2006), learning driver models for adaptive cruise control (Rosenfeld et al. 2012), and improving vehicle power management by estimating road time and traffic congestion conditions (Park et al. 2009). However, although a great deal of effort has gone into the design of the electrical and mechanical aspects of start-stop systems (Bishop et al. 2007; Furushou et al. 2012), reinforcement learning has not (to the best of our knowledge) been applied to optimizing the software start-stop controller prior to our work. However, machine learning has wide applicability to the broader class of energy conservation problems. We cover four example applications here, to illustrate where it can be applied and what methods are used.

Machine Learning for Adaptive Power Management. Machine learning can be used for power conservation by minimizing the power used by a computer, turning off components that have not been used recently, provided the user does not begin using the components again shortly thereafter. Theocharous et al. (2006) applied supervised learning to this area, learning a model of user behavior that classified the current state as a suitable time to turn the components of the laptop off. One notable aspect of their approach was that they created custom contexts and trained custom classifiers for each context. This is an interesting area to explore, especially if the contexts can be determined automatically.

Design, Analysis, and Learning Control of a Fully Actuated Micro Wind Turbine. Another interesting application of machine learning is work by Kolter, Jackowski, and Tedrake (2012), in maximizing the energy production of a wind turbine. The goal is to choose the settings of a wind turbine that maximize energy production. Their work used a sampling based approach to approximate a gradient. This approach is more complex than the Kohl-Stone algorithm, as they form a second order approximation of the objective function in a small area known as the trust region around the current policy. They then solve the problem exactly in the trust region and continue. This approach takes advantage of the second order function information, hopefully allowing for quicker convergence of the policy parameters. The disadvantages of the approach are that it is more complex requires more samples to form the second order approximation. In their case the calculations could be done before the wind turbine was deployed, which meant the increased time for computation was not an issue.

Energy conservation in buildings. Dalamagkidis and Kolokotsa (2008) sought to minimize the power consumption in a building subject to an annoyance penalty that was paid if the temperature in the building was too warm or hot, or if the air quality inside the building was not good enough. Their approach was based on a value function method, which was able to optimize a custom policy that was near the performance of a hand-tuned policy in 4 years of simulator time.

Adaptive Data Centers. The last example of machine learning being used to minimize power is in data centers, which have machines that can be turned off based on the workload in the near future. The goal is to meet the demands of the users of the data center, while keeping as many of the machines turned off as possible. Bodik et al. (2008) wrote about using linear regression to predict the workload that would be demanded in the next two minutes. Based on this prediction the parameters of a policy vector were changed. This resulted in 75% power savings over not turning any machines off, while only having service level violations in .005% of cases.

Summary and Path to Deployment

Start-stop controllers are typically calibrated by through an iterative process of manually adjusting control gains until the control meets a defined set of fuel economy, performance and drivability requirements. By applying machine learning, it is possible to simultaneously optimize these factors and automatically generate a superior calibration—using real data, we were able to improve Ford's existing policy

by approximately 12%. This type of automated optimization is more time-efficient than a conventional manual tuning approach. Since policy search was conducted over the parameters already present in Ford's production start-stop controller, the updated parametrization obtained by policy search can be directly and immediately incorporated into the start-stop controller calibration process. Prior to deployment, the new parametrization would have to go through a verification process, which would typically include an objective verification of performance targets (such are engine restart time under a wide range of conditions including stopping on an incline) and a subjective verification of customer acceptability (possibly through a customer clinic).

This work also shows that the use of multiple policies could further improve the performance of the start-stop controller. Multiple policies could be implemented via manual driver selection (e.g. high sensitivity versus low sensitivity) or through automatic policy selection. Our analysis found that even with just two selectable policies, the better performing policy can change from stopping event to stopping event. We could thereby gain a further 8% improvement if we were able to accurately identify which could be applied to each stop. Our experiments showed that the right policy to apply at each stop could not be determined based on the driver's behavior *before* the stop; however, they suggested that the driver's behavior *during* the stop might be more useful, thus indicating directions for improving the policy used for start-stop control. We conclude that a feed-forward prediction based on the driver behavior immediately leading up to the stopping event would be best suited to this problem. The key challenge in implementing an adaptive start-stop controller will be finding a reliable feed-forward indicator.

The future development of a method for selecting one from a set of possible policies for use online presents a more complicated path to deployment than using a single optimized policy. The two (or more) policies are simply different sets of parameters that can be swapped out of Ford's existing parametrized start-stop controller, but determining the thresholds for switching between policies will need to be calibrated. For instance, a specialized policy might only be selected over a base policy if the likelihood that it will improve performance is greater than 95%. With this approach, an online adaptive start-stop controller can be deployed with minimal risk.

Acknowledgments

This research was funded by a grant from the Ford-MIT Alliance.

References

Bishop, J.; Nedungadi, A.; Ostrowski, G.; Surampudi, B.; Armiroli, P.; and Taspinar, E. 2007. An engine start/stop system for improved fuel economy. SAE Technical Paper 2007-01-1777.

Bishop, C. 2006. *Pattern Recognition and Machine Learning*. New York, NY: Springer.

Bodik, P.; Armbrust, M.; Canini, K.; Fox, A.; Jordan, M.; and Patterson, D. 2008. A case for adaptive datacenters to conserve energy and improve reliability. Technical Report UCB/EECS-2008-127, University of California at Berkeley.

Csepinszky, A. 2011. Operational results and conclusions of the fot execution phase of eurofot european large scale field operational test. In *18th ITS World Congress*.

Dalamagkidis, K., and Kolokotsa, D. 2008. Reinforcement learning for building environmental control. In Weber, C.; Elshaw, M.; and Mayer, N., eds., *Reinforcement Learning, Theory and Applications*. I-Tech Education and Publishing. 283–294.

Ford Motor Company. 2013. Ford motor company 2013 annual report. http://corporate.ford.com/doc/ar2012-2012%20Annual%20Report.pdf.

Furushou, M.; Nishizawa, K.; Iwasaki, T.; and Tahara, M. 2012. Stop-start system with compact motor generator and newly developed direct injection gasoline engine. SAE Technical Paper 2012-01-0410.

Kohl, N., and Stone, P. 2004. Machine learning for fast quadrapedal locomotion. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, 611–616.

Kolter, J.; Jackowski, Z.; and Tedrake, R. 2012. Design, analysis, and learning control of a fully actuated micro wind turbine. In *Proceedings of the American Control Conference*.

Murphey, Y.; Masrur, M.; Chen, Z.; and Zhang, B. 2006. Model-based fault diagnosis in electric drives using machine learning. *IEEE/ASME Transactions on Mechatronics* 11:290–303.

Navigant Research. 2013. Stop-start vehicles. http://www.navigantresearch.com/research/stop-start-vehicles.

Park, J.; Chen, Z.; Kiliaris, L.; Kuang, M.; Masrur, M.; Phillips, A.; and Murphey, Y. 2009. Intelligent vehicle power control based on machine learning of optimal control parameters and prediction of road type and traffic congestion. *IEEE Transactions on Vehicular Technology* 58(9):4741 – 4756.

Prokhorov, D., ed. 2008. *Computational Intelligence in Automative Applications*, volume 132 of *Studies in Computational Intelligence*. Springer.

Rosenfeld, A.; Bareket, Z.; Goldman, C.; Kraus, S.; LeBlanc, D.; and Tsimoni, O. 2012. Learning driver's behavior to improve the acceptance of adaptive cruise control. In *Proceedings of the Twenty-Fourth Innovative Applications of Artificial Intelligence Conference*, 2317–2322.

Sutton, R., and Barto, A. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Sutton, R.; McAllester, D.; Singh, S.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, 1057–1063.

Theocharous, G.; Mannor, S.; Shah, N.; Gandhi, P.; Kveton, B.; Siddiqi, S.; and Yu, C.-H. 2006. Machine learning for adaptive power management. *Intel Technology Journal* 10:299–312.

Vong, C.-M.; Wong, P.-K.; and Li, Y.-P. 2006. Prediction of automative engine power and torque using least squares support vector machines and Bayesian inference. *Engineering Applications of Artificial Intelligence* 19:277–287.

Wang, X.; McGee, R.; and Kuang, M. 2013. Vehicle system control for start-stop powertrains with automatic transmissions. SAE Technical Paper 2013-01-0347.

Williams, R. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8:229–256.