Direct Transcription Optimization for Maximizing Traffic Efficiency

Edward Wang Zhongxia Yan

***Abstract*—In this paper, the problem of traffic control to maximize the overall efficiency of a single road with respect to speed using controls. Previous works by Wu et al [1] have shown that using model-free policy gradient methods to control a single vehicle is able to stabilize the traffic at the optimal but unstable limit cycle of the IDM in a ring network setting. However, recent developments in underactuated control give insight that the idea of traffic control might be achieved using techniques from other aspects of underactuated robotics. We demonstrate a direct-transcription approach to enable all vehicles to maximize the average speed and demonstrate a simple simulation study in a small ring road. Finally, we compare the efficacy of such an "advanced" approach with a classical PID [2] approach and compare their results. Our code can be found on GitHub.**

***Index Terms*—traffic, control, direct transcription, optimization, underactuated, simulation, sumo**

I. Introduction

We would like to apply control methods to the "underactuated" traffic optimization setting. Imagine a simulated traffic system consisting of a road network with vehicles; our goal is to maximize the overall efficiency (e.g. average speed) of all vehicles in the road network by controlling the acceleration of a small subset of the vehicles. Note that the acceleration of the uncontrolled majority of the vehicles is provided by the Intelligent Driver Model (IDM) [3], which is a time-continuous car-following model commonly used to simulate freeway and urban traffic. Previous work by Wu et al [1] have shown that using model-free policy gradient methods to control a single vehicle is able to stabilize the traffic at the optimal but unstable limit cycle of the IDM in a ring network setting. Our project focuses on a similar setting but applies model-based methods to stabilize the optimal limit cycle.

II. Intelligent Driver Model (IDM)

The Intelligent Driver Model models human driving behavior given the speeds and relative positions of a vehicle and the vehicle in front. Let x_i be the position of the vehicle i's front bumper and $v_i = \dot{x}_i$ be the speed of vehicle i. Let $i-1$ be the vehicle directly in front of vehicle i (i.e. $x_{i-1} > x_i$). The IDM specifies acceleration $\dot{v}_i$ as a function of x_i, v_i, x_{i-1}, v_{i-1}, and a set of physically interpretable IDM parameters. Define relative velocity for vehicle i to its preceding vehicle as

$$\Delta v_i = v_i - v_{i-1}$$

and spacing in front of the vehicle i as

$$s_i = x_{i-1} - x_i - \ell_{i-1}$$

where ℓ_{i-1} is the length of vehicle $i-1$. Then the IDM gives

$$s^*(v_i, \Delta v_i) = s_0 + v_i T + \frac{v_i \Delta v_i}{2\sqrt{ab}}$$

$$\dot{v}_i = a\left(1 - \left(\frac{v_i}{v_0}\right)^\delta - \left(\frac{s^*(v_i, \Delta v_i)}{s_i}\right)^2\right) \tag{1}$$

where the parameters are defined as

- v_0 is the velocity that the vehicle would drive in free traffic.
- s_0 is the minimum allowed distance to the car in front.
- T is the "safe time headway," i.e. vehicle i should lag behind vehicle $i-1$ by at least this amount of time.
- a is the maximum vehicle acceleration.
- b is the comfortable braking deceleration.
- δ is usually set to 4.

From the perspective of the controlled set of vehicles, the IDM simulating the rest of the vehicles defines part of the state transition function.

III. Task Description

Our traffic optimization task consists of vehicles $i \in \{1, \ldots, N_v\}$ on a ring-shaped road network with circumference L, where each vehicle $i-1$ precedes vehicle i (i.e. $x_{i-1} > x_i$). To simply our notation, when we specify a vehicle with $i-1$, we always mean $i-1 \mod N_v$. We assume that we can directly control vehicle $i = 1$'s acceleration

$$-1 \le u = \dot{v}_1 \le 1 \tag{2}$$

and $\dot{v}_i$ for $i \in \{2, \ldots, V\}$ are given by Equation (1). For IDM parameters, we use

$$\ell_i = 5,\ v_0 = 30,\ s_0 = 2,\ T = 1,\ a = 1,\ b = 1.5,\ \delta = 4$$

We simulate a discretized version of the task with step size $\mathrm{d}t$ by updating x_i and v_i according to

$$v_i[n+1] = v_i[n] + \dot{v}_i[n]\,\mathrm{d}t \tag{3}$$

$$x_i[n+1] = x_i[n] + \frac{v_i[n] + v_i[n+1]}{2}\,\mathrm{d}t \tag{4}$$

We assume that the initial speeds $v_i[0] = 0,\ \forall i$ and sample initial positions $x_i[0] \sim \mathcal{N}(\frac{i}{N_v}L, 1)$, i.e. the vehicles are initially roughly evenly-spaced in the ring network. We simulate for N steps after the initial step. The objective of controlling vehicle $i = 1$ is to maximize the average vehicle speed while avoiding collisions with other vehicles.

A. IDM Behavior

In the case where all N_v vehicles follow the IDM in the ring network, it is well-known that the state of all vehicles going at the same optimal speed v_f and each consecutive pair of vehicles separated by the same spacing s_f is an unstable limit cycle as shown in Figure 1. As mentioned by [4] and even verified in the real-world human-driver setting [5], any small deviations from this limit cycle results in the formation of backward-propagating stop-and-go waves as shown in Figure 2, which leads to a stable but sub-optimal equilibrium. Therefore, we would like to use control-based methods to mitigate the formation of these stop-and-go waves.

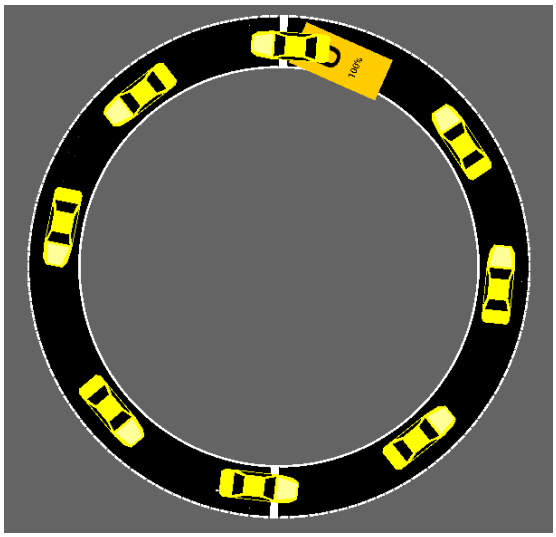


Figure 1: Optimal but unstable limit cycle. Vehicles have the same spacing s_f and constant velocity v_f.

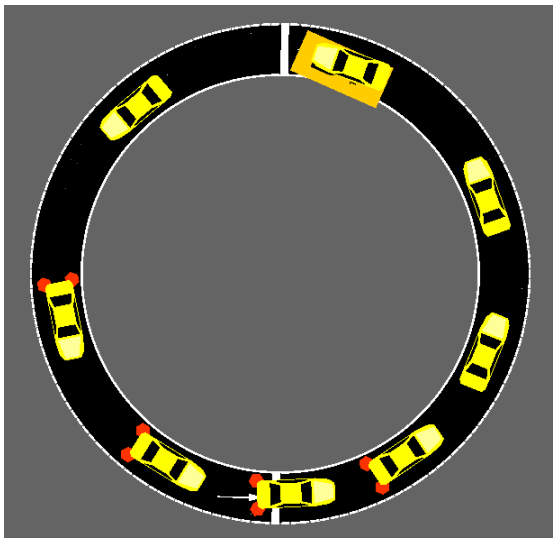


Figure 2: Sub-optimal but stable limit cycle with backward-propagating stop-and-go waves. Vehicles will alternate between slow and fast speeds.

IV. Optimization Formulation

We use a direct-transcription-like approach and formulate the task as an optimization problem with a mean speed maximization objective

$$\min_{x.[\cdot],v.[\cdot]} -\sum_{n=0}^{N}\sum_{i=1}^{N_v} v_i[n] \tag{5}$$

and the following constraints

1) $x.[0]$ equal the initial positions and $v.[0] = 0$.
2) $v.[\cdot] \geq 0$.
3) $x_{i-1}[n] - x_i[n] \geq \ell_{i-1} + s_0$ for $n \in \{1, \dots, N\}$ is the spacing constraint; there must be at least a gap of size s_0 between vehicles.
4) Equation (3) for updating v every step, where for every $n \in \{0, \dots, N-1\}$ we plug in acceleration constraint Equation (2) for $\dot{v}_1[n]$ and IDM Equation (1) for $\dot{v}_i[n]$, $i \in \{2, \dots, N_v\}$. For the latter case, we can get rid of the inverse $\frac{1}{s_i^2}$ from Equation (1) by multiplying both sides by s_i^2 and rearranging.

$$\begin{aligned} &s_i[n] = x_{i-1}[n] - x_i[n] - \ell_{i-1} \\ &s_i[n]^2 \left(v_i[n+1] - v_i[n] - a\,\mathrm{d}t \left(1 - \frac{v_i[n]^\delta}{v_0^\delta} \right) \right) \\ &\quad = a\,\mathrm{d}t \left(s_0 + v_i[n]T + \frac{v_i[n](v_i[n] - v_{i-1}[n])}{2\sqrt{ab}} \right)^2 \end{aligned} \tag{6}$$

   This gets us a nonconvex 6th order (since $\delta = 4$) polynomial constraint.
5) Equation (4) for updating x every $n \in \{0, \dots, N-1\}$.

A. Change of Variable

In addition to the formulation above, we also try changing the position variables $x_i[n]$ into vehicle spacing $s_i[n] = x_{i-1}[n] - x_i[n] - \ell_{i-1}$. Therefore the optimization problem becomes

$$\min_{s.[\cdot],v.[\cdot]} -\sum_{n=0}^{N}\sum_{i=1}^{N_v} v_i[n] \tag{7}$$

such that

1) $s.[0]$ equal the initial spacing and $v.[0] = 0$.
2) $v.[\cdot] \geq 0$.
3) $s_i[n] \geq s_0$ for $n \in \{1, \dots, N\}$ to enforce minimum spacing.
4) $\sum_{i=1}^{N_v}(s_i[n] + \ell_i) = L$ enforces that the total spacing plus the total vehicle lengths is equal to the circumference.
5) Equation (6) for updating v every $n \in \{0, \dots, N-1\}$.
6) We replace the constraint (4) for updating x with

$$\begin{aligned} &s_i[n+1] - s_i[n] \\ &= \tfrac{1}{2}\,\mathrm{d}t(v_{i-1}[n] + v_{i-1}[n+1] - v_i[n] - v_i[n+1]) \end{aligned}$$

B. Optimizing for Unstable Limit Cycle Parameters

We note that if all N_v vehicles use IDM, then we can solve for the parameters of the optimal (but unstable) limit cycle by symmetry. Clearly, since we assume that the vehicle length $\ell. = 5$, the spacing for each vehicle in this limit cycle is given by

$$s_f = \frac{L}{N_v} - \ell$$

The vehicle speeds v_f in this limit cycle are all equal by symmetry and $\dot{v}|_{v=v_f} = 0$, therefore by IDM equation (1)

$$0 = a\left(1 - \left(\frac{v_f}{v_0}\right)^{\delta} - \left(\frac{s_0 + v_f T + 0}{s_f}\right)^2\right)$$

$$0 = 1 - \left(\frac{v_f}{v_0}\right)^{\delta} - \left(\frac{s_0 + v_f T}{s_f}\right)^2$$

and we can easily find v_f by finding the root of this polynomial.

Knowing the optimal parameters (s_f, v_f), we can stabilize the optimal limit cycle by updating our objective to penalize mean square error (MSE) to (s_f, v_f)

$$\min_{s.[\cdot], v.[\cdot]} \sum_{n=0}^{N} \sum_{i=1}^{N_v} (v_i[n] - v_f)^2 + (s_i[n] - s_f)^2 \tag{8}$$

Additionally, we can add further constraints to further assist optimization

1) $v.[\cdot] \leq v_f + 0.25$, since we would like no vehicle's speed to be too much faster than the v_f.
2) $s.[\cdot] \leq 2s_f$, since we would like no vehicle's spacing to be too much bigger than s_f.

Finally, we set the "Guess" for $v.[n]$ and $s.[n]$ for $n \in \{1, \ldots, N\}$ to be v_f and s_f with the hope that this will help the optimizer find a trajectory with the desired behavior.

V. Comparison to Traditional Control

To evaluate the performance of direct transcription optimization, we compare its performance to that of a traditional PID controller [2], as shown below.

$$u(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{i}} \int_0^t e(t')\, dt' + K_{\mathrm{d}} \frac{de(t)}{dt}$$

The traditional PID controller aims to control a particular state variable by minimizing its error $e(t)$, by applying an input $u(t)$ based on 1) a factor K_p directly proportional to the error; 2) a factor K_i proportional to the accumulated error to date; and 3) a factor K_d proportional to the derivative of the error.

In the context of this study, we define the goal of the PID controller for a car as keeping a constant following distance to the car in front of it - $e(t) = s(t) - s_f$, where s_f is the desired following distance. For example, if the car has a goal of being 20m behind the next car but is currently 25m behind, the error $e(t)$ is $25 - 20 = 5$. Likewise, we define the input $u(t)$ as the force/acceleration being applied to the car. For simplicity, we assume that the mass of the car is 1 kg. This is inconsequential since scaling the mass of the car simply scales the PID controller's parameters.

The proportional controller K_p aims to counteract the error by applying force. For example, if the car is lagging behind its expected distance ($e(t) > 0$), the proportional controller will increase the force/acceleration applied to catch up. Likewise, if the car is further ahead than it should be ($e(t) < 0$), then the controller will brake the car to increase the following distance.

The integral controller K_i aims to correct steady-state error and reduce rise time by accounting for the total amount of error that has been taking place. For example, if the error integral is very large, that means that the car has been significantly behind target for a long time, and therefore the integral controller counteracts that by increasing the acceleration. While our SUMO model does not model friction, integral control could also help counteract steady-state error there.

Finally, the derivative controller K_d aims to correct overshoot. For example, if the error is rapidly decreasing, it means that we are approaching our target setpoint and should decrease our control input to avoid overshooting. Likewise, if the error is rapidly increasing, it means that we are diverging from our setpoint and should increase our control input to counteract it. Derivative controllers can be hard to effectively use, since any noise in the error signal $e(t)$ will be amplified upon the taking of a derivative.

VI. Results

We run all experiments on a ring network with circumference $L = 80$ m/s, $N_v = 8$ vehicles, and a simulation step size of $\Delta t = 0.5$ s. Our calculated speed for the optimal limit cycle characterized in Subsection IV-B is $v_f = 3.00$ m/s. We implement the IDM exactly as described here, without any noise (besides for the initial position of the vehicles). We visualize our experiments with the Simulation of Urban MObility (SUMO) traffic simulator [6].

The ring scenario has some contrast that is different from a regular road network. For one, it does not model lane changes, merges, ramps, and other important traffic dynamics. Finally, it can lead to a positive feedback loop where a controlled car that is stalled can lead to cars stalled in front of it, causing the controlled car is further stalled. In contrast, on an long open-ended road, it is not possible for a stalled car to stall itself in this manner.

For the optimization experiments we use the Sparse Nonconvex OPTimizer [7] provided by the PyDrake [8] framework. Due to the nonconvex nature of the optimization and its run-time issues and difficulties finding a solution, we generally solve the optimization for a short number of steps at a time, usually $N \leq 10$. For time efficiency, we then execute the N optimized steps before running optimization again, instead of re-optimizing after each step.

We show the vehicle speeds and optimization time per step of all experiments in Table I.

	N	speed (m/s) mean $\pm$ std	time (s) / step
baseline	N/A	2.42 ± 1.16	0
mean speed	1	2.74 ± 0.85	0.013
mean speed	2	2.37 ± 1.20	0.04
mean speed	5	2.37 ± 1.19	0.48
MSE (s_f, v_f)	1	2.97 ± 0.25	0.016
MSE (s_f, v_f)	5	2.96 ± 0.25	0.18
MSE (s_f, v_f)	10	2.96 ± 0.25	0.63
PID long following	N/A	0.503 ± 0.26	0
PID short following	N/A	1.65 ± 1.34	0

Table I: Performance results for different values of N for the baseline (IDM control policy), maximizing mean speed with $s.[\cdot]$ as variables, and minimizing MSE to (s_f, v_f). Here $v_f = 3.00$ m/s. For the case of maximizing the mean speed, we observe that the SNOPT solver usually wasn't able to find a solution for $N = 2$ and $N = 5$, so we had to resort to using IDM policy for those steps; this probably contributes to the low performance of those results.

A. Baseline: All IDM Vehicles

As mentioned in III-A, controlling all N_v vehicles with IDM results in backward-propagating stop-and-go waves. In the stable limit cycle, the mean vehicle speed is 2.42 m/s, the minimum speed is 0 m/s, and the maximum speed is 4.5 m/s. We show the time-space diagram for the baseline in Subfigure 3a, clearly demonstrating the formation of stop-and-go waves.

B. Trajectory Optimization with Position

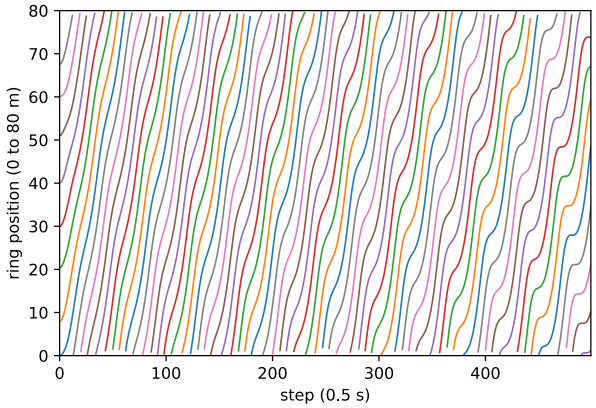
For this method, we perform the optimization as described in Equation (5). We find that in this setting, SNOPT was often not able to find the trajectory.

C. Trajectory Optimization with Spacing

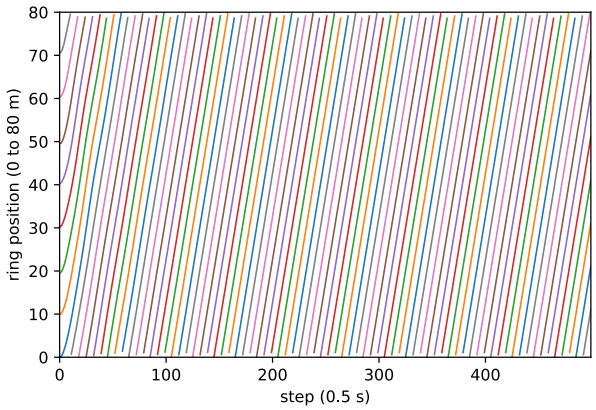
For this method, we perform the optimization as described in Equation (7). We find that in this setting, SNOPT sometimes does not find a trajectory $N > 1$, especially at higher speeds. To handle failures to find a trajectory, we fall back to the IDM controller in those steps, and re-optimize the next time step. When we run trajectory optimization with $N = 1$ we find that the trajectory's behavior is somewhat shortsighted and does not fully stabilize the unstable limit cycle. The mean speed is 2.74 m/s, which is significantly better than the baseline mean speed of 2.42 m/s. For $N > 1$, we find that SNOPT does not find a trajectory too often, leading to IDM-level performances.

D. Trajectory Optimization towards (s_f, v_f)

For this method, we perform the optimization as described in Equation (8). We find that in this setting, SNOPT may find solution for at least $N = 10$. We found that the optimized trajectories for any $N \geq 1$ is able to stabilize the optimal limit cycle. The mean speed here is 2.96 m/s, which is very close to $v_f = 3.00$ m/s. Referring to the time-space diagram in Subfigure 3b, we see that there is almost no congestion.



(a) Baseline: sub-optimal but stable IDM limit cycle with backward-propagating stop-and-go waves. Vehicles will alternate between slow and fast speeds.



(b) Minimizing deviation to (v_f, s_f) stabilizes the unstable limit cycle of IDM.

Figure 3: Time-space diagram: each color represents one of the 8 vehicles. The speed of the vehicle at a particular space and time is given by the slope.

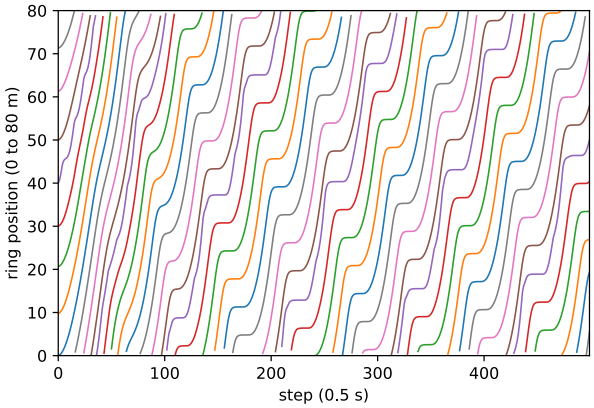
E. PID Controller with Short Following Distance

In Figure 4a, we display the time-space diagram results for a tuned PID controller to follow at a distance of $s_f = 2.2$m. The parameters used were $K_p^+ = 1.6$, $K_p^- = 85.5$, $K_i = 0$, and $K_d = 0$.

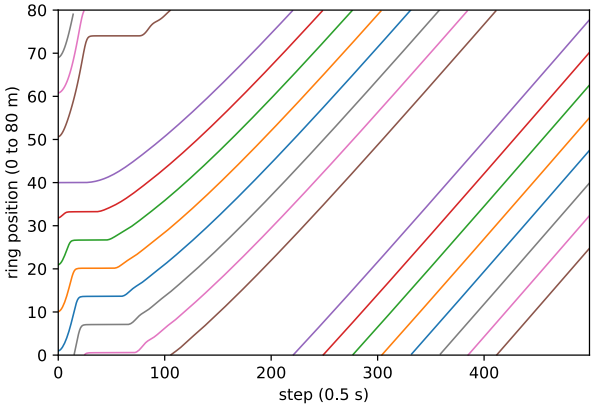
Since the SUMO simulator models traffic collisions by halting the simulation, it required that the PID controller be tuned to prevent collisions. One challenge this led to was that close following required rapid braking in order to avoid a collision, but using the same high-magnitude K_p for acceleration caused the vehicle to crash more often.

As a result, this controller augments the standard PID controller with asymmetric proportional control in order to induce high braking without also inducing high acceleration.

The integral controller is difficult to use since integral windup causes the integral controller to induce unstable os-



(a) PID with short following distance



(b) PID with long following distance

Figure 4: Time-space diagram: each color represents one of the 8 vehicles. The speed of the vehicle at a particular space and time is given by the slope.

cillations of increasing magnitude. For example, when the car is stalled in a traffic wave, the error integral starts to accumulate greatly, resulting a massive burst of acceleration once the leading car is able to move again.

We attempted mitigation of integral windup by resetting the integral term but it did not help as the integral windup would simply return, if not more viciously than before during over-acceleration and stalls.

It might be the case that PID loops are not well tuned for stalling. One overall issue with the simulation setup is that real-life automotive PID loops are not designed for a loop scenario where a car stalling causes the space in front of the car to be blocked. This may cause some of the integral windup weirdness that is not seen in other automotive applications.

In the beginning of the simulation, when the integral term is mostly zero, this was not an issue, but as the amount of time was spent in stalls increased, the car would accelerate more aggressively, which would then increase the amount of stalling in the system and thus increasing the acceleration even more aggressively in a vicious feedback loop.

The derivative controller was not attempted and turned out to be unnecessary because overshoot was not a major consequence observed from the controller.

From a traffic modelling perspective, this mirrors the effect of aggressively tailgating drivers on causing traffic stalls.

F. PID Controller with Long Following Distance

In Figure 4b, we display the time-space diagram results for a tuned PID controller to follow at a distance of $s_f = 22$m. The parameters used were $K_p = 0.006$, $K_i = 0$, and $K_d = 0$.

The PID controller with a longer following distance behaved more smoothly and did not require as aggressive accelerations as the shorter following distance did.

One issue is that longer following distances ran into issues with the spacing available in the ring as a whole. For example, if the PID control is set to a target s_f that is larger than the space available in the ring, the controller responds just by stalling the car.

Overall, a longer following distance leads to more stable. While the integral controller could be tuned and enabled without severe consequences for some values, the same issue of integral windup appears again, so it was ultimately disabled.

The derivative controller was not used in this case for similar reasons as in the case of short following distance.

VII. Feedback Keyword

The keyword is “underactuated”.

VIII. Conclusion

Overall we found that trajectory optimization was easily able to stabilize the trajectory around the unstable limit cycle of the IDM policy. We found that trajectory optimization with the mean speed objective is partially successful, but is overall a harder optimization objective. Our PID controllers were able to stop the stop-and-go waves at the cost of going at a much slower speed, and may need more tuning of the parameters or different formulation to achieve better performance. We note that the we could also achieve higher velocities than v_f by reducing the spacing in front of the vehicle that we control. One difficulty that we had in the traffic optimization case is that the dynamics models of traffic behaviour may be complicated even for small scenarios. Furthermore, the ring scenario lacks some features such as lane changes and intersections, and introduces other dynamics such as self-stalling. Solving some of these additional challenges may involve modeling discrete choices. It would be interesting to look into other work that have considered these factors into the dynamics model. Our code can be found on GitHub.

References

[1] Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M Bayen. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *arXiv preprint arXiv:1710.05465*, 2017.

[2] Bruce Land. The PID controller. 2012.

[3] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805–1824, Aug 2000.

[4] Mauro Garavello and Benedetto Piccoli. *Traffic flow on networks*, volume 1. American institute of mathematical sciences Springfield, 2006.

[5] Yuki Sugiyama, Minoru Fukui, Macoto Kikuchi, Katsuya Hasebe, Akihiro Nakayama, Katsuhiro Nishinari, Shin ichi Tadaki, and Satoshi Yukawa. Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam. *New Journal of Physics*, 10(3):033001, mar 2008.

[6] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, December 2012.

[7] Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

[8] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019.