Autonomous Driving in a Roundabout With Rule-Breaking Humans

Emiko Soroka
Department of Aeronautics & Astronautics
Stanford University
esoroka@stanford.edu

***Abstract*—While intersection are often used as a test case for interaction in autonomous driving, a roundabout is another interesting case where multiple agents interact. In this paper we design a discrete MDP to represent traversing a roundabout with other agents, learn the optimal policy for navigating a roundabout, and investigate how it changes when some agents break the rules of the road.**

I. Introduction

Roundabouts are relatively uncommon compared to intersections in the US, but are rising in popularity due to their safety and environmental benefits [3]. Studies have found that replacing signalized intersections with roundabouts improves throughput for low-speed intersections [4] and reduces the risk of rear-end collisions [5]. One empirical study conducted in the US found that replacing intersections with roundabouts effected a 40% reduction in crashes, with an even greater decrease in traffic injuries and deaths [4].

Although roundabouts are simple to navigate, with the cardinal rule being that traffic entering the roundabout yields to traffic already in the circle, human drivers who are unaware of this rule often hold up traffic by stopping in the roundabout. We are interested in two questions:

- Can we train an RL agent to successfully navigate a roundabout?
- How do rule-breaking drivers affect the optimal policy for navigating the roundabout?

Problem Description

We represent this problem as a Markov decision process (MDP) over discrete states and actions.

States

We define three integer-valued states: s_1, s_2 and s_3 (Fig. 1).

- s_1 has 30 values, corresponding to the rounded distance traveled along the lane.
- s_2 is the distance between the ego vehicle and the closest vehicle that presents a collision risk. This distance ranges from 0 to 10.
- s_3 is the ego vehicle velocity, sorted into buckets from 0 to 10.

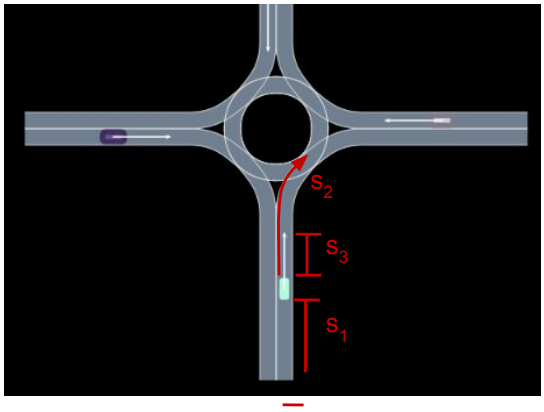

Fig. 1. The three discretized state space variables.

Actions

Our RL agent has three actions available: `ZERO` (no acceleration action), `ACCEL` and `DECEL`. Lane-keeping is managed separately to simplify the problem, and no lane changes take place.

With this representation, there are $|S_1| \times |S_2| \times |S_3| = 3630$ states and 3 actions. This is a relatively small problem, so it is possible to learn an optimal policy by a direct method such as Bellman iteration.

Observations

We assume there is no noise in the system and the states are directly observed.

Reward

The RL agent receives a time-based reward $R(t)$ (Eq. (1)) for reaching its goal on the other side of the roundabout, where N is the number of time steps in the simulation and t is the step where the goal is reached. This incentivizes crossing the roundabout quickly. If a collision occurs, the agent receives a reward of -1000 and the simulation stops.

$$R(t) = \begin{cases} 10 + (N - t) & \text{for reaching the end at step } t \\ 0.1\Delta s & \text{for traveling } \Delta s \text{ in } N \text{ steps} \\ -1000 & \text{for collisions.} \end{cases} \tag{1}$$

Modeling Other Drivers

To provide other agents for the simulation, we defined two fixed policies by observing human drivers at the Manzanita Field roundabout.

Rule-Following Driver Model

The rule-following driver model approximates the correct way to navigate a roundabout. When approaching, it yields to traffic already in the roundabout. If another vehicle is too close, it decelerates unless said vehicle is not currently in the roundabout. Otherwise, it prefers to maintain a constant speed.

This model can be represented by the code:

```
function get_lawful_action(model)
    # If distance to another car is < 2
    # and we are NOT in the roundabout
    if model.s2 < 2 &&
       (model.s1 < L || model.s1 > 2*L)
        return :DECEL
    # maintain speed
    elseif model.s3 >= 10
        return :ZERO
    # speed up to desired speed
    else
        return :ACCEL
    end
end
```

Rule-Breaking Driver Model

The rule-breaking driver model behaves like the safe driver model with probability $p \in [0, 1]$. With probability $1 - p$ it may take a rule-breaking action such as:

- Yielding to a vehicle waiting to enter the roundabout.
- Not yielding to another vehicle when entering the roundabout.

This model can be represented by the code:

```
function get_lawless_action(model)
    # Should we yield when entering??
    if rand() < model.p && model.s2 < 2 &&
      (model.s1 == 1 || model.s1 == 3)
        return :DECEL
    # Should we stop in the roundabout??
    elseif rand() > model.p && model.s2 < 2
        returj :DECEL
    # stay at desired speed
    elseif model.s3 >= 10
        return :ZERO
    # speed up if too slow
    else
        return :ACCEL
    end
end
```

Note: As becomes clear later, these two agents are not particularly skilled at navigating the intersection.

II. Methodology

We used `AutomotiveSimulator.jl` [1] to define the environment, rule-breaking and rule-following agents, and the RL agent itself. We used online Bellman iteration [2] to learn the roundabout policy by running the simulation with varying starting positions and other agents. We studied two cases:

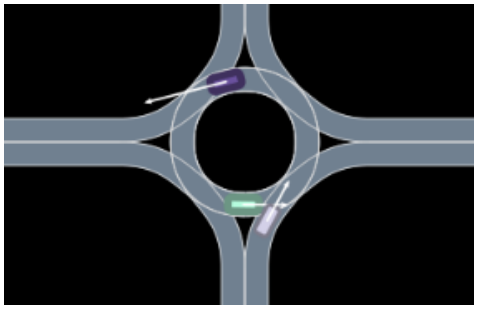

Fig. 2. The unsafe driver (gray) enters the roundabout in front of the rule-following green car, causing a collision.

All safe drivers:

The other drivers all follow the rules, yielding and proceeding correctly. Since Bellman iteration is exact, we expect this to converge to an optimal policy for navigating the roundabout.

Some unsafe drivers:

A fraction $r \in (0, 1)$ of the drivers break the rules with probability p. We used $p = 0.5$, $r = 0.25$ and $r = 0.5$.

Exploration:

We initially tested several different methods of balancing exploration vs exploitation and settled on a constant 50% chance of taking a random action during training.

Other approaches tested included:

- Greedy selection: this did not work because the agent did not explore enough of the state space and never learned to move forward.
- Decreasing the exploration over time: this turned out to be unnecessary.

III. Implementation

Most of the work on this project was in coming up with a reasonable discretization of the state space and debugging issues related to reward and simulation design.

State Space and Reward

The space was chosen to be large enough to capture the desired behavior while still being tractable, and various modifications had to be made to resolve problems that arose during training. For example, we initially tried to use three states to capture position: before, in, and after the roundabout. This failed because the RL agent would take an action and remain in the same state, so it wasn't able to train.

We also tested several reward functions before settling on an appropriate one - some tests only gave a positive reward for crossing the roundabout, but the agent wasn't able to reach it before the simulation ended and again, didn't train effectively.

Simulation Design

A final major issue was preparing simulations that effectively explored the state space. Initial simulations used three other agents, randomly assigned to each lane, but this resulted in very few observed states where a collision was possible. The resulting agent was unlikely to decelerate and avoid collisions. We increased this number to 5, ensuring that the agent encounters situations with and without another car sharing its lane.

Collision Course Detection

A major challenge was defining and fine-tuning the human driver models for the RL agent to train with, especially correctly implementing the yielding behavior. Many challenges also arose in identifying whether two vehicles are on a collision course. Two examples of this situation are shown in Figure 3.

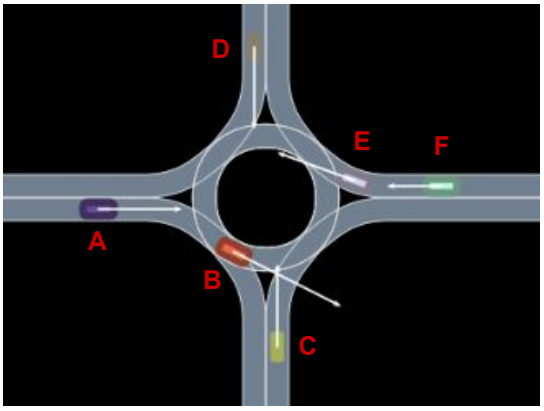

Fig. 3. Vehicle F is on a collision course with E. C is on a collision course with B, but B is not on a collision course with C (because C should yield).

The collision course detection was eventually implemented with a special case for the roundabout.

```
function on_coll_course(v1::Entity, v2::Entity)
    # check projection of velocity vectors
    vel1 = velg(v1.state)
    dx = (posg(v2.state) .- posg(v1.state))[1:2]
    projection = dot(dx, vel1)
    if projection > 5.0 && projection < 150
        return true
    end
    # SPECIAL CASE: if we are not in
    # the roundabout and the other vehicle is
    s1 = posf(v1.state).s
    s2 = posf(v2.state).s
    if (s1 < L || s1 > 2L) &&
       (s2 >= L && s2 <= 2L)
        return true
    end
    return false
end
```

IV. Results

Table **??** shows the reward obtained over an average of 10 trials for each trained Rl agent. The trials were sorted into three categories based on outcome: completed trials (the agent successfully crossed the roundabout), passed trials (didn't cross, but didn't collide) and collisions.

The first three entries are for the hand-coded rule-following agent to provide a baseline. We see that the agent is imperfect (likely due to simplifications made in its design), as even when all agents follow the rules, one collision was observed.

As expected, the number of collisions increases as the percentage of unsafe drivers increases. Interestingly, for successful runs, the completed score is very similar for agents trained with only safe drivers and agents trained with some unsafe drivers.

Our hypothesis was that unsafe drivers would result in a slower policy that receives less reward, however this doesn't seem to be the case. This could be due to a bug (especially in the function that determines whether two vehicles are at risk of colliding) or a poorly designed reward function.

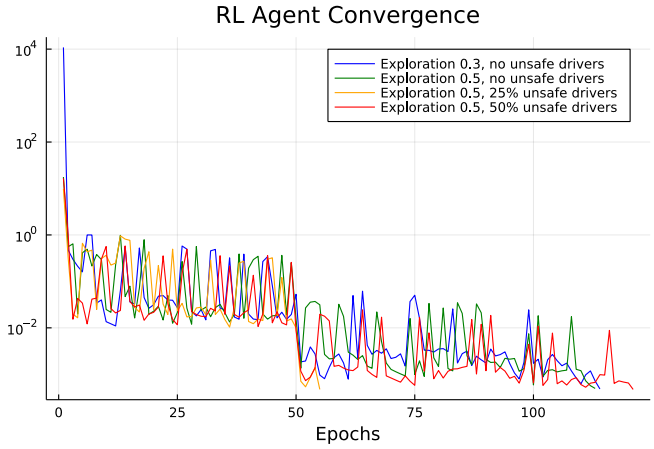

Fig. 4. Convergence of four test cases for the final RL agent. The jump is where we re-started the training loop after pausing it to save a snapshot of the value function.

We used 50 runs per epoch, checking the convergence condition $\|U_k - U_{k-1}\|_2 / \|U_k\|_2 < \epsilon$ at the end of each epoch. For all agents, ϵ was set to $5e^{-4}$.

The addition of rule-breaking drivers decreased the convergence rate (Figure 4). However, all four agents were able to converge to a reasonably effective policy.

Evaluation of RL Agents

We trained four agents in total. To investigate the effect of the exploration rate p (the probability of taking a random action), we trained one agent with $p = 0.3$ and one with $p = 0.5$, both in a simulation with no rule-breaking drivers. We found that the agent with the lower exploration rate performed noticeably worse, so we selected $p = 0.5$ for the remaining agents.

The third agent was trained with $p = 0.5$ and 20% rule-breaking drivers in the roundabout (4 rule-following drivers and one rule-breaking). The last agent was trained with $p = 0.5$ and 40% rule-breaking drivers (2 rule-breaking, 3 rule-following). This policy took the longest time to converge.

With 4 vehicles in the roundabout, the rule-breaking and rule-following baselines performed best (Fig. 5). The rule-breaking agent, in particular, never stopped to wait for another vehicle in the roundabout (Fig. 6). We weren't able to draw any conclusion about whether the agents trained with rule-breaking drivers were more cautious, resulting in fewer successful crossings and fewer collisions, from Figure 6.

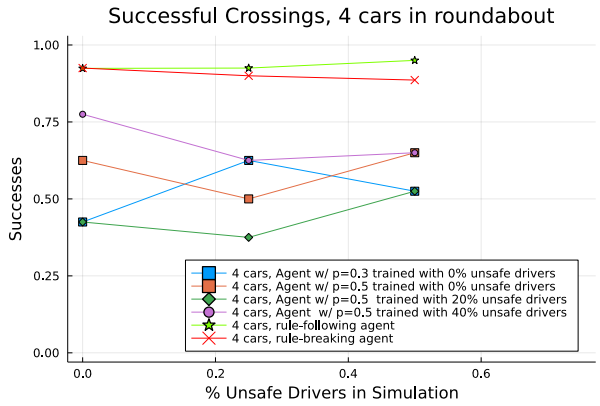

Fig. 5. Success rate with 4 cars in roundabout.

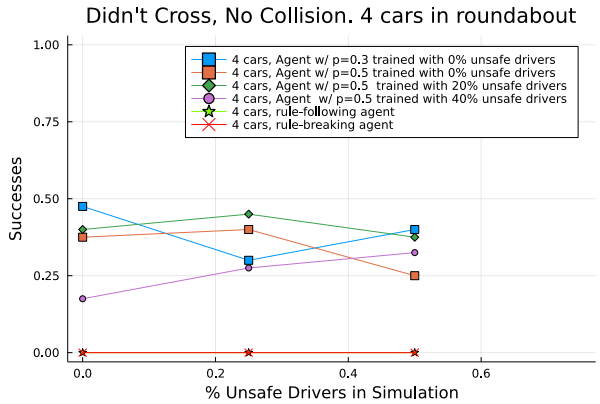

Fig. 6. Pass rate with 4 cars in roundabout.

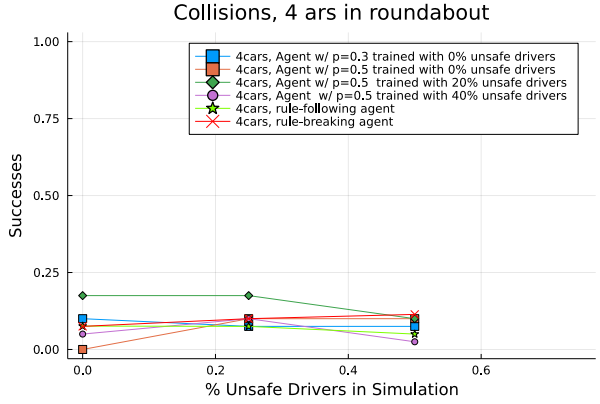

Fig. 7. Collision rate with 4 cars in roundabout.

With 6 vehicles in the roundabout, performance was very poor with many collisions (Fig. 10) and few successful attempts. All of the agents performed similarly poorly at crossing (Fig. 8 and 9), which raises the question of how well the hand-coded agents are in the first place. Perhaps given better drivers to interact with, the RL agents would learn a more effective policy.

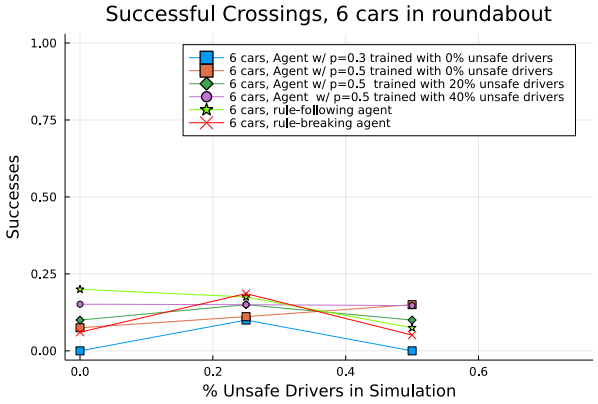

Fig. 8. Success rate with 6 cars in roundabout.

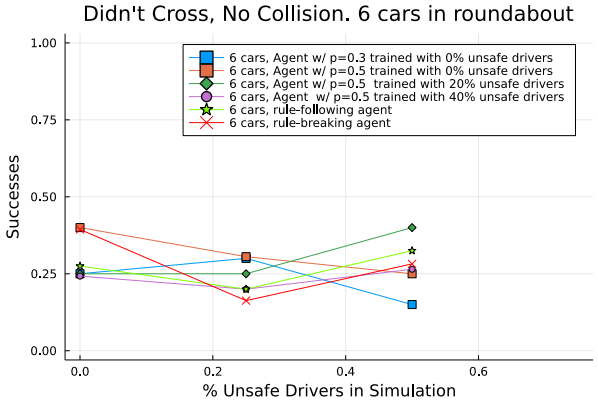

Fig. 9. Pass rate with 6 cars in roundabout.

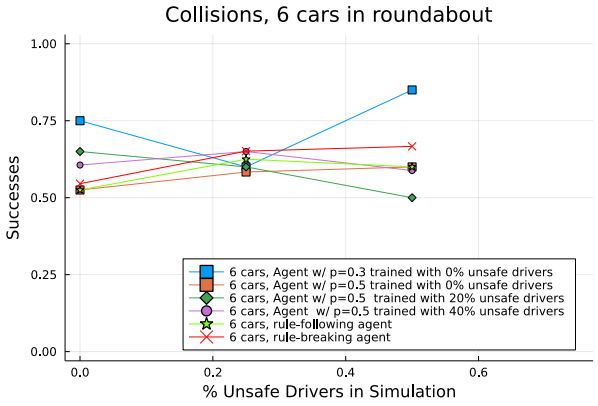

Fig. 10. Collision rate with 6 cars in roundabout.

V. Did RL actually work?

We would be remiss if we did not evaluate the RL agents skeptically, as RL has been accused of many misdeeds. Since this project suffered from poor implementation of the rule-breaking and rule-following drivers, we compared the state-action maps of the RL agents against the rule-following driver.

Conclusions

Source code, data and figures for this paper can be accessed at.

Acknowledgments

I would like to thank Prof. Kochenderfer and all of his TAs for being great this quarter, teaching a lot of interesting material and giving us the freedom to be creative with our final projects.

SISL developed the AutomotiveSimulator.jl package which provided a reliable base for RL training and simulations. All errors in software judgment are my own.

Finally, my original proposal was to use MDPs for modeling intersections. Credit for the switch to a more interesting topic goes to all of the sub-optimal drivers holding up traffic in Stanford's roundabouts.

References

[1] Automotivesimulator.jl, 2022. URL https://sisl.github.io/AutomotiveSimulator.jl/dev/.

[2] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray. *Algorithms for decision making*. MIT press, 2022.

[3] S. Mandavilli, M. J. Rys, and E. R. Russell. Environmental impact of modern roundabouts. *International Journal of Industrial Ergonomics*, 38(2):135–142, 2008. ISSN 0169-8141. doi: https://doi.org/10.1016/j.ergon.2006.11.003. URL https://www.sciencedirect.com/science/article/pii/S0169814106002526. Spanning the Gap from Traditional Ergonomics to Health and Safety Issues.

[4] B. N. Persaud, R. A. Retting, P. E. Garder, and D. Lord. Safety effect of roundabout conversions in the united states: Empirical bayes observational before-after study. *Transportation Research Record*, 1751(1):1–8, 2001.

[5] F. F. Saccomanno, F. Cunto, G. Guido, and A. Vitale. Comparing safety at signalized intersections and roundabouts using simulated rear-end conflicts. *Transportation Research Record*, 2078(1):90–95, 2008.