A TRAFFIC MODELS

A.1 Multi-step microscopic model

The pseudo-code of the model used for the experiments is listed in Algorithm 1. Some of the design choices are noteworthy:

- When an agent chooses its next edge, it does so using the *anticipated latency* of the network, i.e., it calculates the latency of an edge e as $l_e(f_e + 1)$, since the flow on the edge will be incremented as soon as the car enters the edge. There are two ways to use the anticipated latency for finding a shortest path: (a) Only use the anticipated latency for the first edge, or use it for all edges along the route. Moreover, to account for the total cost, each agent uses its value-of-time and value-of-money properties and chooses a shortest path with respect to the weighted sum of anticipated travel time and anticipated tolls.
- The speed of an agent along an edge (i.e., the number of steps it takes until it reaches the end of the edge) is set when the agent enters the edge, and is not changed anymore, even if the latency of the edge changes in subsequent steps when the agent is still traversing the edge. This gives a more realistic behavior than having incoming agents change the speed to agents which are further along the edge.
- At each step, the agents move along their current edge according to their speed. As soon as the end of the edge is reached, the agent stops, and at the next step makes its choice about the subsequent edge. This means that in this model, it always takes at least one step to traverse an edge, even if the latency is less than 1. Accordingly, the latency values need to be large enough to allow for differentiated travel times under this discretization.
- When an agent reaches its destination, the position is reset to the source. This step is executed prior to deciding on the subsequent edges and progressing traffic. The approach facilitates constant flow with a fixed number of agents on the graph.

Algorithm 1: Multi-step traffic simulation

```
input: Network, cars, number of steps
foreach step do
    Compute flow for each edge in network based on car
     positions and update network attributes accordingly;
    foreach car c do
        Move c based on speed;
        if c has reached the end of its edge e then
            Decrement flow on e;
        end
    end
    foreach car c do
        if c has reached its target then
            Reset c;
        end
    end
    foreach car c (in random order) do
        if c is at a node then
            Determine next edge e for c;
            Set c's speed based on latency of e ;
            Increment flow on e;
        end
    end
end
```

A.2 Single-step microscopic model

For the examples in Section 4, we use a model in which each agent is a distinct entity; however, agents do not move on the network, but instead choose their entire route at once. The travel times and total costs are then computed in a single step. To avoid spikes and oscillations, we still let the agents choose their routes sequentially in random order (observing what the previous agents have chosen at the same step). From a game-theoretic viewpoint, this setup corresponds to an Extensive-Form Game, while the multi-step setup in Appendix A.1 is a Stochastic Game. Its results are "cleaner" than in the multi-step model, since an agent can be thought of as a continuous flow which is, at the same time, located at each edge of its route.

B DOUBLE BRAESS PARADOX

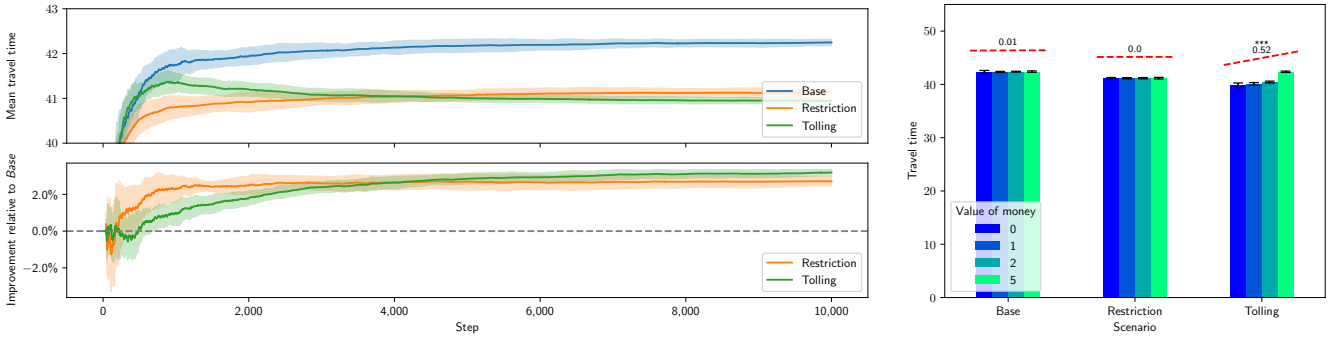
For this "Double Braess Paradox", we first observe that, in the original network (Figure 3a), there are three types of latency functions: (a) *High constant latencies* like edge $(0, 2)$, (b) *affine latencies* like $(0, 1)$ and (c) *low constant latencies* like $(1, 2)$. The latter type of edge is removed to resolve the paradox.

Therefore, we re-use the high-constant-latency edge $(0, 2)$ as the *low-constant-latency* edge in another, superimposed paradox, such that its removal improves the flow on this second structure. Accordingly, we add two nodes A and B (taking the roles of 0 and 3 in the original setting), and connect them to the existing nodes 0 and 2. The latency functions of the new edges need to be chosen such that the unique optimal route from A to B is $A - 0 - 2 - B$, regardless of the traffic rate.

Note that the edge $(0, 2)$ now has a double role: To resolve the paradox on the graph $\{0, 1, 2, 3\}$, it needs to be present; to resolve the paradox on the extended graph $\{A, 0, 2, B\}$, it needs to be removed. Hence, we now have demand-dependent optimal restrictions as intended.

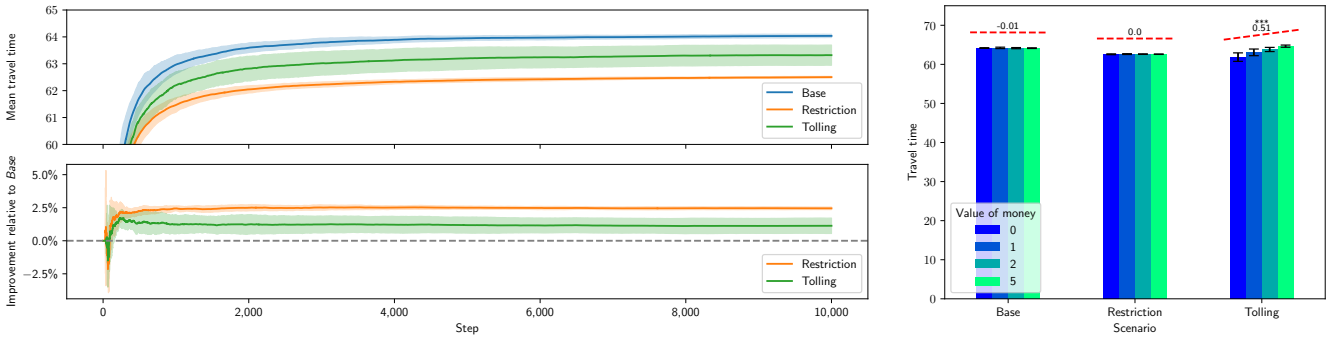
C $G_{n,p}$ GRAPHS

Our experiments follow the formal notation of [48] in which each graph G is drawn from the standard Erdős-Rényi model $G_{n,p}$. Each edge is given an independent affine latency function $l_e(n) = a + b(\frac{n}{c})$ where a, b and c are integers drawn independently from three fixed uniform distributions $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$. We search for Braess' Paradox by repeatedly sampling graphs, traffic rates, source nodes s, and destination nodes d. Hereby, agents are initialized on random edges of the simple paths between s and d. The restriction effect is measured by independently removing each edge e with a high equilibrium flow from G. We find Braess' Paradoxes when generating traffic scenarios with the following parameters and seed 46:



(a) $G_{50,0.07}$: **Social welfare (i.e., mean travel time)**

(b) $G_{50,0.07}$: **Fairness**



(c) $G_{34,0.06}$: **Social welfare (i.e., mean travel time)**

(d) $G_{34,0.06}$: **Fairness**

Figure 8: Additional results of the $G_{n,p}$ graph experiment. Similar to the results shown in the main paper (Figure 5), both *Restriction* and *Tolling* improve the mean travel time compared to *Base*. However, value of money significantly influences travel time with *Tolling* ($p \leq 0.01$).

	$\mathcal{A}$	$\mathcal{B}$	C	f	s	d	e
$G_{50,0.07}$	$\mathcal{U}_{\{3,4\}}$	$\mathcal{U}_{\{2,3\}}$	$\mathcal{U}_{\{1,3\}}$	38	23	33	(23, 30)
$G_{34,0.06}$	$\mathcal{U}_{\{3,4\}}$	$\mathcal{U}_{\{2,4\}}$	$\mathcal{U}_{\{1,4\}}$	38	6	20	(25, 9)
$G_{59,0.08}$	$\mathcal{U}_{\{2,4\}}$	$\mathcal{U}_{\{2,3\}}$	$\{3\}$	31	3	27	(20, 41)

The travel time and fairness metrics for $G_{50,0.07}$ and $G_{34,0.06}$ are shown in figure 8.

D GENERALIZED BRAESS GRAPHS

The same reasoning as in Appendix B applies to the generalized Braess graphs: In every iteration i, the high-constant-latency edge (s_{i-1}, w) becomes a low-constant-latency edge in the subgraph $\{s_i, s_{i-1}, w, t_i\}$.

Formally, the graph $B_n = (V_n, E_n)$ is defined as:

$$
\begin{aligned}
V_n &:= \{s_0, ..., s_n, v, w, t_0, ..., t_n\} \\
E_n &:= \{(s_0, v), (v, w), (v, t_0)\} \cup \{(s_1, s_0), ..., (s_n, s_{n-1})\} \cup \\
&\quad \{(s_0, w), ..., (s_n, w)\} \cup \{(s_0, t_1), ..., (s_{n-1}, t_n)\} \cup \\
&\quad \{(w, t_0), ..., (w, t_n)\}
\end{aligned}
$$

with

$$
l_e(n) := \begin{cases} 1 & \text{if } e = (v, w) \\ 2 + 6 \cdot \frac{n}{c} & \text{if } e = (s_i, s_{i-1}) \text{ or } (s_0, v) \text{ or } (w, t_i) \\ 11 + 2i & \text{if } e = (s_i, w) \text{ or } (s_i, t_{i+1}) \end{cases} .
$$

It is clear that edges involving nodes s_j or t_j with $j > i$ are irrelevant for traffic flowing from s_i to t_i since there are no directed paths going through these nodes. For commodity (s_i, t_i), the optimal restriction is closing edges $\{(s_k, w), 0 \leq k \leq i - 1\}$.

E SEEDS

We have used the following seeds for our experiments:

Experiment	Seeds
$G_{n,p}$	41, 42, 43, 44, 45, 46, 47, 48, 49, 50
Braess	42, 43, 44, 45, 46

F PARKING SCENARIO

In [15]'s dynamic pricing scenario (see Figure 9), parking spaces are available at a number of different places within a city district. A number of cars (agents) with individual targets on the map navigate the road network in order to find a parking space. For each agent, the utility of finding a parking space depends on various factors

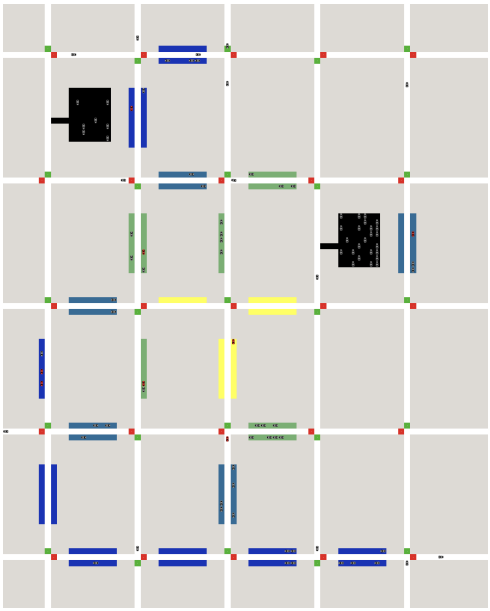


Figure 9: Grid network for dynamic pricing of parking spaces as proposed by [15].

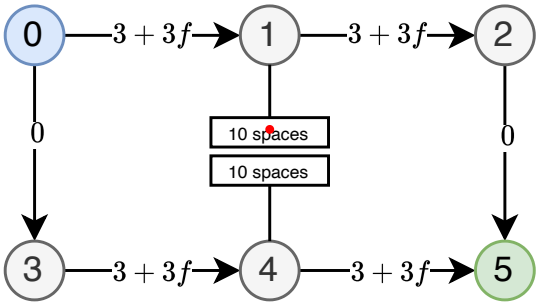


Figure 10: Parking environment with two parking lots which can be reached from nodes 1 and 4, respectively. The red dot denotes the center of the normal distribution of targets, meaning that lot 1 is more attractive on average.

such as the travel time, the distance from the parking space to the target, or the agent's preference to park roadside or in a garage.

We simplify this scenario in the following way, as illustrated in Figure 10: Using a directed graph with latency functions as in our main experiments, we assign two-dimensional positions to all nodes. Moreover, parking lots are located at certain locations on the map and connected to nodes, such that an agent can use a parking space when it has reached the respective node. Cars have uniformly random entry and exit points (nodes) and normally distributed targets (coordinates) within the limits of the grid. The utility of an agent is a weighted sum (where the weights correspond to the conversion factor between driving and walking) of travel time between start node and parking space and distance between parking space and target.

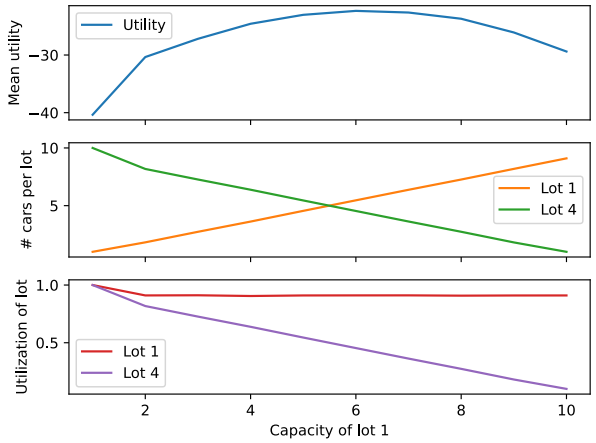


Figure 11: Social welfare of restricted parking for different numbers of parking spaces at lot 1. When all 10 spaces are open, route $0 - 1 - 2 - 5$ is congestion-prone since lot 1 is the better option for most agents. Demand can be balanced by closing some of the spaces at lot 1, leading to optimal social welfare when 4 spaces are closed.

Let us use the simple grid shown in Figure 10, and assume that there are two parking lots which are accessible from nodes 1 and 4, respectively. Crucially, the center of the agents' target distribution is $(0.5, 0.7)$, meaning that lot 1 is slightly more attractive than lot 4.

As the parking management authority, we can only set the number of parking spaces at lot 1; at lot 4 there are always 10 spaces. Similarly to Braess' Paradox, closing some of the parking spaces can increase social welfare: Simulating the scenario for all values $k \in [0, 10]$ gives the social welfare graph in Figure 11. We can see that the social welfare takes its maximum at $k = 6$, while both extreme values of k are detrimental.

This phenomenon can be explained as follows: When there are 10 parking spaces at lot 1, most cars decide to go there, resulting in over-utilization and congestion of edge $(0, 1)$. When there are fewer parking spaces, the lot becomes gradually less attractive, until the incentives are sufficiently balanced to allow for even traffic on both $(0, 1)$ and $(3, 4)$.

While we do have restrictions in this scenario, they are not aimed at removing specific actions from the agents' action spaces. Instead, the agents can still choose the same actions (i.e., use the same edges and nodes), but the utility of doing so—in RL terms, their action-value functions—are affected by the restrictions.