Roundabout Design Optimization: Lane Count and Diameter Effects under Symmetric Demand

Midterm Report

Massi Afzal, Jaathavan Ranjanathan, and Bach Vu

December 2, 2025

GitHub Project Repository

Contents

1 Introduction

This project develops a configurable, text-based microsimulation (in python) of a four-arm roundabout to quantify how design and behavioural parameters affect operational efficiency. The simulator is simple, no graphics and single file, so that geometry (e.g., diameter, number of circulating lanes) and behaviour (e.g., arrival rates, driver reaction time, critical gaps) can be varied easily and their impacts measured reliably and reproducibly. The current focus is the roundabout and maximizing its efficiency, as well as identifying breaking points in the system (e.g., the critical volume at which another lane or a different sized diameter becomes necessary)

In parallel, we have replicated these scenarios in SUMO (Simulation of Urban Mobility), using the same modelling logic, comparable queuing rules at entries, consistent car-following dynamics, and aligned demand patterns, so results are methodologically comparable across platforms. Coding our own Python simulation gives us greater control and transparency over assumptions, algorithms, and metrics used, allowing for a better understanding of how each modelling choice influences outcomes. SUMO, in turn, provides a visual environment, richer built-in diagnostics/metrics, and a great ecosystem for scenario management. We have used SUMO both to augment our analysis (e.g., additional KPIs and visual inspection of flows) and to validate the Python results, treating agreement across platforms as evidence for model correctness and having used any discrepancies to refine assumptions (e.g., gap-acceptance thresholds, reaction delays, lane-discipline rules). Together, the Python and SUMO tracks create a robust workflow for optimizing designs and testing sensitivity to demand, geometry, and driver behaviour.

2 Problem Statement & Implementation Challenges

2.1 Core Problem

Urban traffic intersections operate as critical bottlenecks in transportation networks. Poor design or control strategies lead to:

- **Capacity saturation**: Queue divergence when demand exceeds service capacity
- **Excessive delays**: Reduced level-of-service affecting user satisfaction
- **Safety concerns**: Increased conflict points and crash risk
- **Environmental impact**: Elevated emissions from idling and stop-and-go behaviour

Fundamental question: Given demand characteristics (volume, turning movements) and site constraints, what intersection control strategy and geometric configuration maximize efficiency while ensuring stability?

2.2 Technical Challenges

2.2.1 Stochastic Variability

Traffic is inherently random. To simulate realistic conditions, we must model and consider: **arrival processes** (how drivers arrive), **driver heterogeneity** (how drivers behave), **turning choices** (how will drivers move through the intersection), **behavioural uncertainty** (reaction times, desired speeds, aggressiveness).

2.2.2 Continuous Dynamics with Discrete Events

Vehicles follow continuous ordinary differential equations (car-following, lane-changing) while experiencing discrete events (arrivals, merges, exits).

2.2.3 Reaction Delay

Human reaction time (τ) is not instantaneous. Therefore, the equation used to model the simulation must consider that acceleration at time t depends on states at $t - \tau$.

2.2.4 Gap Acceptance at Yield Points

Modelling roundabout entries require considering:

- **Critical gap** T_c: Is there a large enough time gap in the circulating traffic for the first vehicle to merge?
- **Follow-up headway** T_f: After the first vehicle merges, can the next vehicles follow with a shorter time gap and enter as a small group (known as a platoon)?
- **Group behaviour**: If an approaching ring vehicle arrives too soon, it can break the platoon, so the next vehicle must wait for a new critical gap T_c.

3 Mathematical & Computational Background

What follows are the various mathematical and computational backgrounds needed to realize this project and to tackle the aforementioned challenges. We'll also demonstrate how we've implemented some of it in our python code, thus far, and how SUMO handles it as well.

Roundabout Modelling & Simulation

3.1 Poisson Arrival Process (Roundabout)

The roundabout vehicle arrival process (upstream and downstream) can be modelled by a homogeneous Poisson process. The one variable (a customizable knob) associated with this model is the rate lambda (vehicles/second). There are two different views:

3.1.1 Count/Macroscopic View

The number of arrivals $N(T)$ in interval $[0, T]$ is Poisson-distributed:

$$P(N(T) = k) = \frac{(\lambda T)^k e^{-\lambda T}}{k!}$$

Properties:

- Mean: $\mathbb{E}[N(T)] = \lambda T$
- Variance: $\text{Var}[N(T)] = \lambda T$ (variability matches mean)
- Gives us the "sometimes a few more, sometimes a few less" effect

3.1.2 Gap/Microscopic View (Interarrivals)

Time between successive vehicles $\Delta t \sim \text{Exponential}(\lambda)$:

$$f(\Delta t) = \lambda e^{-\lambda \Delta t}, \quad \Delta t \geq 0$$

Properties:

- **Memoryless**: $P(\Delta t > t+s \mid \Delta t > t) = P(\Delta t > s)$ — after waiting t seconds with no arrival, the probability of waiting an additional s seconds is unchanged
- **Average interarrival headway**: $\mathbb{E}[\Delta t] = 1/\lambda$
- **Short gaps are common**; long gaps occur but with exponentially decreasing probability

Implementation (Python):

```python
def _next_arrival_time(self, arm: int, now: float) -> float:
    lam = max(1e-9, self.cfg.demand.arrivals[arm])  # lambda avoids zero division
    return now + random.expovariate(lam)            # exp. interarrival
```

Implementation (SUMO): Route files specify Poisson departures via exponentially distributed `depart` times.

3.2 Turning Movement Selection

Each vehicle selects Right/Through/Left with probabilities (p_R, p_T, p_L) summing to 1. Sampling from such a categorical distribution is commonly done with an inverse-CDF trick: draw a uniform distribution $u \in [0, 1)$ and see in which cumulative bucket u falls.

Implementation (Python):

```python
def _draw_turn_steps(self) -> int:
    L, T, R = self.cfg.demand.turning_LTR
    u = random.random()
    if u < R:       return 1    # Right
    elif u < R + T: return 2    # Through
    else:           return 3    # Left
```

Implementation (SUMO): Routes define turning movements with `<route probability="p">` attributes, where SUMO internally samples using the same categorical distribution method.

3.3 Gap Acceptance Model

At a yield line, the first queued vehicle must find a sufficiently large **critical time gap** T_c in the circulating stream to merge. Once that leader enters, subsequent vehicles can follow with a tighter **follow-up headway** T_f, creating a short platoon. This two-stage process reflects realistic driver behaviour at roundabout entries.

3.3.1 Critical Gap (First Vehicle)

Lognormal Distribution $T_c \sim \text{LogNormal}(\mu, \sigma^2)$:

$$f(t) = \frac{1}{t\sigma\sqrt{2\pi}} \exp\left(-\frac{(\ln t - \mu)^2}{2\sigma^2}\right), \quad t > 0$$

Parameter conversion from mean m and standard deviation s:

$$\mu = \ln\left(\frac{m^2}{\sqrt{s^2+m^2}}\right)$$
$$\sigma^2 = \ln\left(1+\frac{s^2}{m^2}\right)$$

Typical values: $m \approx 3.0$s, $s \approx 0.6$s
Rationale: Lognormal produces positive, right-skewed gaps matching empirical data—most drivers accept 2–4s gaps; a minority demand much larger gaps.

3.3.2 Follow-Up Headway (Platooning)

Normal Distribution $T_f \sim \mathcal{N}(\mu_f, \sigma_f^2)$:

$$f(t) = \frac{1}{\sigma_f\sqrt{2\pi}} \exp\left(-\frac{(t-\mu_f)^2}{2\sigma_f^2}\right)$$

Typical values: $\mu_f \approx 2.0$s, $\sigma_f \approx 0.4$s (bounded below at 0.2s for safety)
Rationale: Follow-up headways are more symmetric around a typical value; normal distribution (with safety floor) provides simple approximation.

3.3.3 Merge Conditions

A vehicle merges from queue to ring when **both** conditions hold simultaneously:

Time Condition Time until next circulating vehicle reaches entry point, t_{next}, must exceed required headway:

$$t_{\text{next}} \geq T_{\text{needed}} = \begin{cases} T_c & \text{(first vehicle or broken platoon)} \\ T_f & \text{(platooning)} \end{cases}$$

Space Condition Immediate front and rear spatial buffers at merge point exceed minimum safe distance $s_{\min}$ (typically 5–10m for comfort/safety).

Platoon Disruption If an approaching ring vehicle will arrive before the needed headway materializes ($t_{\text{next}} < T_{\text{needed}}$), the developing opportunity is spoiled and the platoon is broken. Consequently, the next merge attempt reverts to first-car behaviour and requires a new critical gap T_c, so the queued vehicle must wait for that gap before attempting to enter.

Drawing Critical Gap and Follow-Up Headway (Python):

```python
def _draw_crit_gap(self) -> float:
    """Draw critical gap T_c from lognormal distribution"""
    m, s = self.cfg.gaps.crit_gap_mean, self.cfg.gaps.crit_gap_sd
    mu = math.log((m*m) / math.sqrt(s*s + m*m))
    sigma = math.sqrt(math.log(1 + (s*s)/(m*m)))
    return max(0.2, random.lognormvariate(mu, sigma))

def _draw_followup(self) -> float:
    """Draw follow-up headway T_f from normal distribution"""
    m, s = self.cfg.gaps.followup_mean, self.cfg.gaps.followup_sd
    return max(0.2, random.gauss(m, s))
```

Merge Decision Logic (Python):

```python
# Determine required headway (platoon-aware)
needed = self.next_needed_headway[i] or q[0].crit_gap
# T_needed = T_f (platooning) or T_c (first car / broken platoon)

# Compute time until next circulating vehicle reaches entry
t_next = self._time_to_nearest_ring_vehicle(self.entry_pos[i], lane_idx)

# Check time and space conditions
if (t_next >= needed and self._space_ok_at_merge(lane_idx, self.entry_pos[i],
    self.cfg.driver.s0 + 2.0)):
    # Merge accepted: vehicle enters ring
    # Set follow-up headway for next vehicle (platooning)
    self.next_needed_headway[i] = v.followup
else:
    # Check for platoon disruption
    if t_next < needed:
        # Approaching ring vehicle spoils opportunity
        self.next_needed_headway[i] = 0.0  # Break platoon
        # Next merge attempt reverts to first-car behaviour (T_c)
```

Implementation (SUMO): Gap acceptance uses `jmTimegapMinor` for critical gaps and follow-up headways, with an impatience model where accepted gaps decrease with wait time.

3.4 Delay-Differential Equations (DDEs)

Human drivers don't react instantaneously. A simple way to incorporate this is a delay-differential-equation (DDE) surrogate: compute the ego's (specific vehicle currently being simulated) acceleration using the states from $t - \tau$ (ego's own speed and the leader's speed/gap as of τ seconds ago). This can damp unrealistic "instant" reactions and produce smoother, more realistic dynamics.

3.4.1 Informal DDE Representation

$$a(t) = f(\text{ego}(t - \tau), \text{leader}(t - \tau))$$

Implementation (Python):

```python
# Keep ~tau/dt snapshots of (pos,speed) for all vehicles
self.max_hist_len = int(math.ceil(max(0.1, cfg.driver.tau) / cfg.dt)) + 2

# On each time step
self._push_history()  # Save current (pos,speed) by vehicle id

# When computing IDM acceleration
steps_back = int(round(self.cfg.driver.tau / dt))
snap = self._snapshot(steps_back)  # Retrieve states from t-tau
pos_d, v_d = snap.get(v.id, (v.pos, v.speed))  # Ego at t-tau
_, gap_d, vL_d = self._leader_at_delayed_time(lane, pos_d, snap)  # Leader at t-
    tau
```

Implementation (SUMO): `actionStepLength="1.0"` means vehicles update control every 1.0s, approximating $\tau = 1.0$s delay (not a true DDE but emulates similar functionality). The vehicles maintain constant acceleration between action steps.

3.5 Intelligent Driver Model (IDM)

Continuous car-following model computing acceleration based on:

- Own speed v
- Desired speed v_0
- Gap to leader s
- Speed difference $\Delta v = v - v_L$

3.5.1 IDM Equation

$$a = a_{\max}\left[1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*}{s}\right)^2\right]$$

where **desired dynamical distance**:

$$s^* = s_0 + vT + \frac{v\Delta v}{2\sqrt{a_{\max}b}}$$

Parameters:

- s_0: minimum gap (m) — typically 2.0m
- T: desired time headway (s) — typically 1.5s
- $a_{\max}$: maximum acceleration (m/s^2) — typically 2.0 m/s^2
- b: comfortable deceleration (m/s^2) — typically 3.0 m/s^2
- δ: acceleration exponent — typically 4

Properties:

- Free-flow: When $s \to \infty$, $a \to a_{\max}[1 - (v/v_0)^{\delta}]$ (accelerate toward v_0)
- Interaction: When $s \approx s^*$, second term activates smooth deceleration
- Emergency braking: If leader suddenly stops, b term provides strong braking

The continuous dynamics $\dot{v} = a$ and $\dot{x} = v$ are advanced using the **forward Euler method** with time step Δt:

$$v(t + \Delta t) = \text{clamp}\left(v(t) + a(t) \cdot \Delta t,\ 0,\ v_0\right)$$
$$x(t + \Delta t) = \left[x(t) + v(t + \Delta t) \cdot \Delta t\right] \bmod C$$

where C is the ring circumference (positions wrap around). Speed is clamped to $[0, v_0]$ for validity.
Typical Δt: 0.1s (Python), 1.0s (SUMO's default simulation step)
Implementation (Python):

```python
# IDM acceleration calculation using delayed states
s0, T, a_max, b, delta = drv.s0, drv.T, drv.a_max, drv.b_comf, drv.delta
dv = v_d - vL_d  # Speed difference (ego - leader) at t-tau
s_star = s0 + v_d*T + (v_d*dv) / max(1e-6, 2.0*math.sqrt(a_max*b))
acc = a_max * (1.0 - (v_d / max(0.1, v0))**delta
               - (s_star / max(1e-3, gap_d))**2)

# Euler integration with speed clamping
v_new = clamp(v.speed + acc * dt, 0.0, v0)
x_new = (v.pos + v_new * dt) % self.C  # Modulo for circular ring
```

3.6 SUMO-Specific Computational Models

Although SUMO's default car-following model is the **Krauss model**, our study requires a direct comparison with the Intelligent Driver Model (IDM) used in our Python microsimulation. To achieve this, SUMO implements an improved version of the IDM (sometimes called the EIDM) that modifies the free acceleration behaviour to yield more realistic gaps in homogeneous traffic.

3.6.1 Improved Intelligent Driver Model (EIDM)

The classic IDM, introduced by Treiber et al. (2000), is defined by five key parameters: the desired time headway T, the maximum acceleration $a_{\max}$, the desired deceleration b, the minimum gap s_0, and the acceleration exponent δ. Its desired gap is given by:

$$s^*_{n-1}(t) = s_0 + \max\left(0,\ v_{n-1}(t)\cdot T - \frac{v_{n-1}(t)\left(v_n(t) - v_{n-1}(t)\right)}{2\sqrt{a_{\max}\cdot b}}\right)$$

and its acceleration by

$$a_{\mathrm{IDM}}(t+\Delta t) = a_{\max}\left[1 - \left(\frac{v_{n-1}(t)}{v_0(t)}\right)^{\delta} - \left(\frac{s^*_{n-1}(t)}{s(t)}\right)^2\right].$$

However, it was observed that the classic IDM tends to induce overly large gaps in free traffic. In response, the Improved IDM (EIDM) adjusts the free acceleration term by first defining

$$a_{\mathrm{free}}(t) = a_{\max}\left[1 - \left(\frac{v_{n-1}(t)}{v_0(t)}\right)^{\delta}\right]$$

and then computing the acceleration as follows:

$$a(t+\Delta t) = \begin{cases} a_{\max}\left[1 - \left(\dfrac{s^*_{n-1}(t)}{s(t)}\right)^2\right], & \text{if } s^*_{n-1}(t) \geq s(t), \\ a_{\mathrm{free}}(t)\left[1 - \left(\dfrac{s^*_{n-1}(t)}{s(t)}\right)^{\frac{2a_{\max}}{|a_{\mathrm{free}}(t)|}}\right], & \text{otherwise.} \end{cases}$$

This improved formulation smooths the transition in free acceleration, ensuring that vehicles can more reliably reach their desired speeds without generating excessive gaps.

Parameter Settings and Comparison: To enable a fair comparison with our Python simulation (which employs the classic IDM with delay approximations), SUMO's EIDM was configured with the following settings:

- **Model Selection:** `carFollowModel="IDM"`
- **Critical Gap Parameters:** $s_0 = 2.0$ m and $T = 1.5$ s
- **Dynamic Parameters:** $a_{\max} = 2.0$ m/s^2, $b = 3.0$ m/s^2, and $\delta = 4$
- **Time Step:** `actionStepLength` is set to 1.0 s.

Usage and Comparison: In our text-based simulation the classic IDM, with its continuous time dynamics and delay-differential approximation, governs vehicle behaviour. In contrast, SUMO employs the improved IDM (EIDM) in a discrete-time framework with enhanced free acceleration dynamics.

3.6.2 Junction Model (Priority Rules)

Roundabout priorities defined via `<connection>` priorities in `.net.xml`:

- Ring lanes: high priority (no yield)
- Approach lanes: low priority (must yield to ring)

Gap acceptance (impatience model):

$$t_{\text{accept}} = \text{jmTimegapMinor} \times \left[1 - \min\left(1, \frac{\text{wait\_time}}{\text{jmIgnoreFoeSpeed}}\right)\right]$$

As wait time increases, accepted gap shrinks (driver becomes impatient).

3.7 1-D Ring Geometry (Directed Distance & Wrap)

Concept (general):

- Positions are represented as an arc-length $s \in [0, C)$ on a loop of circumference C.
- The forward (ahead) distance from position a to b is defined by:

$$d = \begin{cases} b - a, & \text{if } b \geq a, \\ b - a + C, & \text{if } b < a, \end{cases} \qquad \text{equivalently } d = (b - a) \bmod C \in [0, C).$$

Implementation (Python):

```python
def aheaddistance(a: float, b: float, C: float) -> float:
    d = b - a
    if d < 0.0:
        d += C
    return d
```

(We already use wrapping for positions via modulo C in the state update.)

3.8 Lateral-Acceleration Speed Cap (Curve Limit)

On a circle of radius R, the lateral acceleration is given by

$$a_{\text{lat}} = \frac{v^2}{R}.$$

Imposing a comfort/safety bound $a_{\text{lat}} \leq a_{\text{lat,max}}$ yields a curvature-limited maximum speed

$$v_{\max} = \sqrt{a_{\text{lat,max}} \times R}, \qquad \text{with } R = \frac{D}{2} \text{ (ring diameter } D\text{)}.$$

Implementation (Python):

```python
def ring_vmax(self) -> float:
    R = max(0.1, self.diameter / 2.0)
    return math.sqrt(max(0.1, self.a_lat_max) * R)

# Apply the cap to the ring desired speed:
self.vmax_ring = min(cfg.driver.v0_ring, cfg.geo.ring_vmax())
```

Implementation (SUMO): Enforce the curve limit by setting the speed limit of the ring edge (`speed`) to $v_{\max}$ (or lower) and, if desired, combine with a driver `speedFactor` for inter-driver variability.

3.9 Optimization Methods

Two complementary approaches for identifying optimal intersection configurations:

3.9.1 Grid Search

Systematic evaluation of discrete parameter combinations. For roundabouts: 3 diameters $\times$ 2 lane configurations $\times$ 5 demand levels = 30 scenarios. Guarantees finding the best configuration within the tested grid, with full visibility of the performance landscape.

Linear Regression for Failure Detection Within grid search, linear regression identifies unstable configurations by analyzing queue length trends over time. For each simulation run, fit:

$$Q(t) = \beta_0 + \beta_1 t + \varepsilon$$

where $Q(t)$ is queue length at time t, β_1 is the slope (growth rate), and ε is residual error. A configuration fails if:

- $\beta_1 > 0.5$ vehicles/minute (persistent queue growth)
- $R^2 > 0.8$ (strong linear trend, not random fluctuation)

This distinguishes true capacity exceedance (steady growth) from temporary congestion (random spikes).

3.9.2 Bayesian Optimization

Intelligent search using Gaussian Process (GP) surrogate models to approximate the performance function $f(\mathbf{x})$ where $\mathbf{x}$ represents parameters (diameter, lanes, demand). The GP model guides sampling via an acquisition function (e.g., Expected Improvement) that balances exploration of uncertain regions with exploitation of promising areas. Converges to near-optimal solutions in 20–50 evaluations vs. 100+ for fine-grained grids, enabling continuous parameter optimization.

4 Methodology

4.1 Python Microsimulation Architecture

4.1.1 Core Simulation Loop

Algorithm 1 Python Microsimulation Main Loop

Initialize:
  Create ring lanes (1 or 2 circular lanes)
  Schedule initial arrivals (Poisson) at each approach
  Initialize history buffer for DDE

for each time step $t = 0, \Delta t, 2\Delta t, \ldots, T_{\text{sim}}$ **do**
  Process arrivals: spawn vehicles at approach heads
  Update all vehicles:
    a. Retrieve delayed states $(t - \tau)$ from history buffer
    b. Compute IDM acceleration using delayed gap/speed
    c. Integrate: $v(t + \Delta t)$, $x(t + \Delta t)$ via Euler
    d. Check for lane changes (if applicable)
  Attempt merges at yield lines:
    a. Check time/space gaps against T_c or T_f
    b. Move vehicle from queue to ring if accepted
  Process exits: remove vehicles at destination exits
  Record metrics: queue lengths, speeds, positions
  Push current states to history buffer
end for

4.1.2 Key Data Structures

- **Vehicle class**: ID, position, speed, lane, turn_steps, crit_gap, followup, etc.
- **Lane containers**: List of vehicles sorted by position
- **Queue containers**: FIFO list per approach
- **History buffer**: Deque of snapshots {vehicle_id $\rightarrow$ (pos, speed)}

4.2 SUMO Pipeline Architecture

The SUMO implementation provides a complete pipeline for network generation, demand simulation, execution, analysis, optimization, and visualization.

Network Generation: Programmatically creates `.net.xml` files from geometric parameters using SUMO's `netconvert` tool. Intermediate XML files define nodes (junction center + approach endpoints), edges (ring arcs + approach/exit roads), and connections (yield priorities). Ring edges form 8 circular arcs with speed limits based on lateral acceleration ($v_{\max} = \sqrt{a_{\text{lat}} \cdot R}$).

Demand and Execution: Generates `.rou.xml` files with Poisson arrival processes ($\lambda \cdot \Delta t$ insertion probability) and runs SUMO via TraCI, collecting real-time metrics (throughput, delay, queue lengths, P95 delay, emissions) in 5-minute windows.

Analysis and Failure Detection: Post-processes simulation data to detect failures using three criteria: (1) queue divergence (linear regression slope > 0.5), (2) capacity saturation (throughput

$> 0.95\times$ theoretical capacity), and (3) excessive delays (mean > 60s or P95 > 120s).
Optimization: Automates parameter sweeps using grid search across geometry (3 diameters $\times$ 2 lane configurations) and demand (5 multipliers), totaling 30 scenarios. Multi-objective ranking identifies configurations optimizing throughput, delay, emissions, and balanced performance.
Visualization: Generates static plots (throughput/delay curves, performance scatter plots, failure boundaries) using Matplotlib/Seaborn.

4.3 Parameter Mapping Strategy

Ensuring equivalence between Python and SUMO implementations:

Table 1: Cross-Platform Parameter Mapping

Concept	Python Implementation	SUMO Implementation	Notes
Arrival Process	`random.expovariate(`λ`)`	`<flow probability="`$\lambda \cdot \Delta t$`">`	Exact equivalence
Turning Choice	Inverse-CDF on U[0,1)	`<route probability="p">`	Exact equivalence
Car-Following	IDM with DDE history	`carFollowModel="IDM"` + `actionStepLength`	Delay approximation
Critical Gap	`random.lognormvariate(`μ, σ`)`	`jmTimegapMinor=3.0`	Mean matching
Follow-Up	`random.gauss(`μ, σ`)`	`jmDriveAfterRedTime=2.0`	Mean matching
Impatience	Not implemented	`jmIgnoreFoeProb` growth	SUMO-specific
Speed Limit	Lateral accel constraint	`speed` attribute on edges	Consistent formula
Lane-Changing	Simple keep-right logic	LC2013 model	SUMO more complex

5 Assessment & Evaluation

5.1 Key Performance Indicators (KPIs)

5.1.1 Primary Metrics

1. **Throughput (veh/hr)**
   - **Definition**: Number of vehicles exiting the roundabout per hour
   - **Goal**: Maximize while maintaining stability
2. **Mean Delays and P95 Delays (s/veh)**
   - **Definition**: Mean delay is the average delay experienced by all vehicles, while P95 delay is the delay threshold exceeded by only 5% of vehicles.
   - **Goal**: Reduce worst-case user experience
3. **Queue Length (vehicles)**
   - **Definition**: Peak number of vehicles waiting at any single approach
   - **Goal**: Stay within geometric capacity (approach_length / avg_vehicle_length)
   - **Failure Indicator**: Queue spillback to upstream intersections

6 Results

6.1 Simulation Setup

All reported experiments use a single 4-arm roundabout with symmetric demand and dynamic gap-acceptance and lane choice. For each configuration, we report:

- **Throughput** (veh/hr exiting the system),
- **Average delay** (s/veh),
- **95th-percentile delay** (P95, s/veh),
- **Maximum queue length** per arm (vehicles).

The text-based Python microsimulation tested arrival rates per arm of

$$\lambda \in \{0.05,\ 0.07,\ 0.10\} \text{ veh/s/arm},$$

corresponding to total demands of approximately

$$720,\ 1008,\ 1440 \text{ veh/hr (all arms combined)}.$$

The SUMO microsimulation tested a broader range:

$$\lambda \in \{0.05,\ 0.10,\ 0.15,\ 0.20,\ 0.25\} \text{ veh/s/arm},$$

corresponding to total demands of approximately

$$720,\ 1440,\ 2160,\ 2880,\ 3600 \text{ veh/hr (all arms combined)}.$$

While demand was varied, diameter was held constant at 45m for all tests.

Both SUMO and the text based simulation evaluated three diameters (30m, 40m, 50m) for each configuration, while demand was held cosntant at 0.10 veh/s/arm (for each arm). Each scenario was simulated for one hour of operation.

6.2 One-Lane Roundabout

6.2.1 Text-Based Simulation

Table 2 summarizes the performance of the 1-lane design:

Table 2: Performance of 1-lane roundabout for increasing demand

λ (veh/s/arm)	Total demand	Throughput	Avg delay	P95 delay	Max queue/arm
0.05	720	≈ 719	≈ 24 s	≈ 116 s	$[7, 6, 12, 12]$
0.07	1008	≈ 791	≈ 318 s	≈ 925 s	$[34, 63, 45, 91]$
0.10	1440	≈ 764	≈ 718 s	≈ 1506 s	$[183, 150, 189, 140]$

At $\lambda = 0.05$ veh/s/arm the 1-lane design serves almost all demand but already exhibits noticeable delay. Between 0.05 and 0.07, throughput increases only slightly, while both average and P95 delay grow by an order of magnitude and queues become very long. At $\lambda = 0.10$ veh/s/arm the system is clearly oversaturated: demand significantly exceeds capacity, queues diverge, and typical delays reach 12–25 minutes.

6.2.2 SUMO Simulation

Table 3 summarizes the SUMO performance of the 1-lane design:

Table 3: SUMO performance of 1-lane roundabout for increasing demand

λ (veh/s/arm)	Total demand	Throughput	Avg delay	P95 delay	Max queue/arm
0.05	720	≈ 780	≈ 2.2 s	≈ 9.6 s	3
0.10	1440	≈ 1140	≈ 8.8 s	≈ 54.6 s	38
0.15	2160	≈ 1128	≈ 43.9 s	≈ 133.5 s	72
0.20	2880	≈ 1296	≈ 42.7 s	≈ 195.0 s	129
0.25	3600	≈ 1320	≈ 71.9 s	≈ 166.9 s	169

At $\lambda = 0.05$ veh/s/arm the 1-lane design serves almost all demand with minimal delay. At $\lambda = 0.10$ veh/s/arm, throughput peaks around 1,140 veh/hr but delays begin escalating significantly. Beyond $\lambda = 0.15$, throughput plateaus or declines (classic congestion where excessive demand causes gridlock). The system is clearly oversaturated at higher demands: queues diverge and delays become catastrophic. The results confirm that 1-lane roundabouts reach practical capacity around 1,140 veh/hr before breaking down.

6.3 Two-Lane Roundabout

6.3.1 Text-Based Simulation

Table 4 shows the corresponding metrics for the 2-lane design:

Table 4: Performance of 2-lane roundabout for increasing demand

λ (veh/s/arm)	Total demand	Throughput	Avg delay	P95 delay	Max queue/arm
0.05	720	≈ 721	≈ 1.8 s	≈ 7.4 s	$[3, 2, 4, 3]$
0.07	1008	≈ 999	≈ 4.3 s	≈ 17.4 s	$[6, 4, 10, 9]$
0.10	1440	≈ 1414	≈ 26.0 s	≈ 131.2 s	$[14, 27, 19, 14]$

Up to 1440 veh/hr of total demand, the 2-lane roundabout does not break down: throughput almost matches demand at all three rates, while queues remain modest and delays stay under roughly half a minute on average even at the highest tested demand. Relative to the 1-lane case, 2 lanes roughly *double* usable capacity and dramatically reduce delay and queue lengths.

6.3.2 SUMO Simulation

Table 5 summarizes the SUMO performance of the 2-lane design:

Table 5: SUMO performance of 2-lane roundabout for increasing demand

λ (veh/s/arm)	Total demand	Throughput	Avg delay	P95 delay	Max queue/arm
0.05	720	≈ 780	≈ 0.6 s	≈ 3.0 s	2
0.10	1440	≈ 1260	≈ 2.2 s	≈ 9.5 s	5
0.15	2160	≈ 1812	≈ 5.1 s	≈ 23.7 s	41
0.20	2880	≈ 2172	≈ 13.6 s	≈ 120.1 s	55
0.25	3600	≈ 2244	≈ 30.5 s	≈ 99.3 s	98

The 2-lane configuration exhibits stable performance up to $\lambda = 0.20$ veh/s/arm, achieving peak throughput around 2,172 veh/hr with manageable queues and delays. At $\lambda = 0.10$ veh/s/arm, throughput reaches 1,260 veh/hr with minimal delays. At $\lambda = 0.25$ veh/s/arm, throughput begins plateauing at 2,244 veh/hr, indicating the approach to practical capacity. The results demonstrate that 2-lane roundabouts roughly double the usable capacity compared to 1-lane designs while maintaining acceptable service quality below saturation.

6.4 Three-Lane Roundabout

6.4.1 Text-Based Simulation

The 3-lane configuration was evaluated under the same three base demands (Table 6):

Table 6: Performance of 3-lane roundabout for increasing demand

λ (veh/s/arm)	Total demand	Throughput	Avg delay	P95 delay	Max queue/arm
0.05	720	≈ 722	≈ 1.3 s	≈ 5.4 s	≈ 3 veh/arm
0.07	1008	≈ 998	≈ 2.2 s	≈ 7.4 s	$[2, 4, 9, 3]$ (max 9)
0.10	1440	≈ 1417	≈ 9.8 s	≈ 67.1 s	$[14, 20, 15, 16]$ (max 20)

For the tested demands, the 3-lane roundabout shows no capacity issues: throughput essentially equals demand at all three values and queues remain short. Delays are even lower than in the 2-lane case at $\lambda = 0.10$ veh/s/arm, especially for the worst 5% of vehicles.

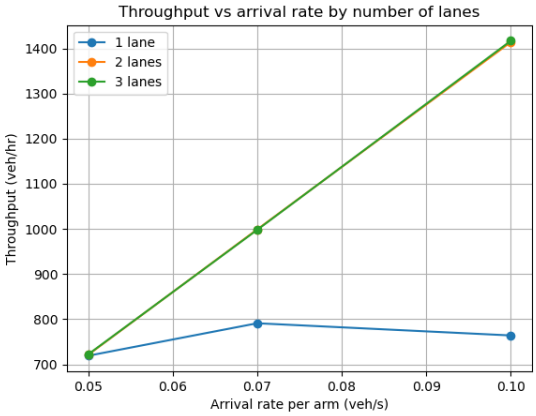
6.4.2 SUMO Simulation

SUMO reveals 3-lane configurations maintain efficiency longest, with practical capacity exceeding 2,180 veh/hr (Table 7):

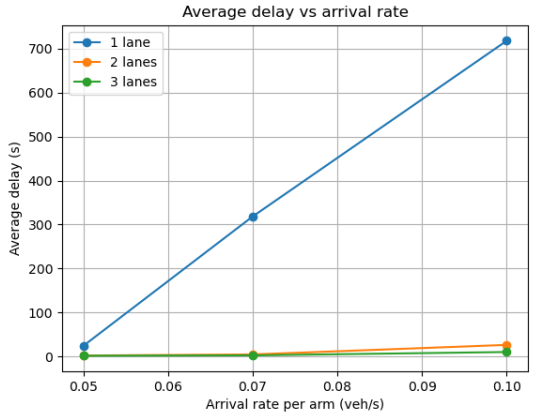
Table 7: SUMO performance of 3-lane roundabout for increasing demand

λ (veh/s/arm)	Total demand	Throughput	Avg delay	P95 delay	Max queue/arm
0.05	720	≈ 768	≈ 0.5 s	≈ 3.5 s	2
0.10	1440	≈ 1248	≈ 1.2 s	≈ 6.1 s	4
0.15	2160	≈ 1884	≈ 4.6 s	≈ 20.2 s	9
0.20	2880	≈ 2268	≈ 4.7 s	≈ 31.0 s	58
0.25	3600	≈ 2772	≈ 8.2 s	≈ 24.4 s	91

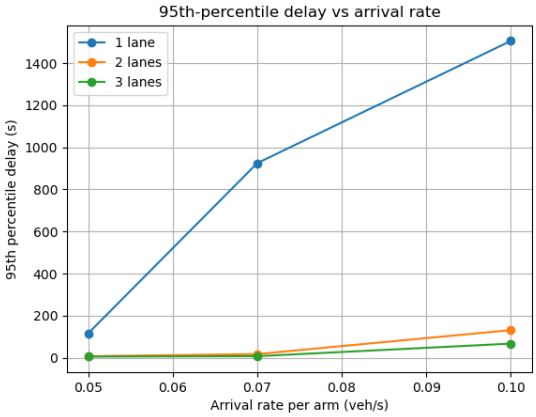
The 3-lane configuration achieves 2,772 veh/hr at $\lambda = 0.25$ veh/s/arm with only 8.2s average delay (lower than 2-lane at $\lambda = 0.20$ veh/s/arm). This demonstrates the third lane's primary value is massive delay reductions with moderate throughput gains. Even at extreme demand, the configuration shows no breakdown signs, suggesting capacity extends beyond 3,000 veh/hr. Queue lengths for 2-lane and 3-lane at $\lambda = 0.25$ veh/s/arm are similar (98 vs. 91 vehicles), indicating entry capacity becomes the bottleneck regardless of circulation lanes at very high demands.



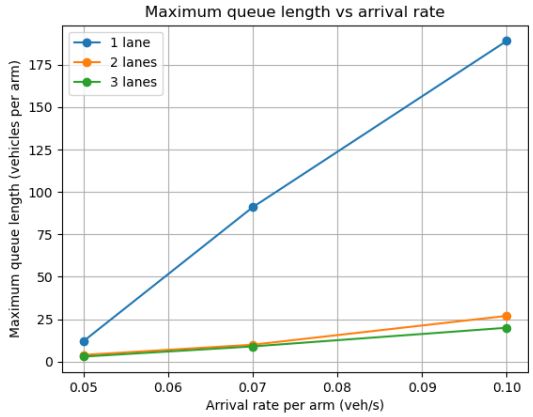
(a) Throughput vs. arrival rate (λ)



(b) Average delay vs. arrival rate (λ)

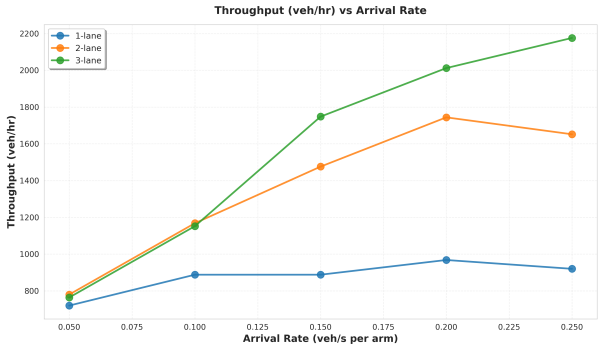


(c) 95th-percentile delay vs. arrival rate

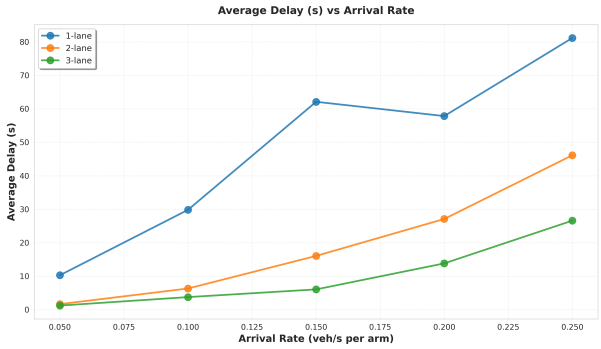


(d) Maximum queue length vs. arrival rate

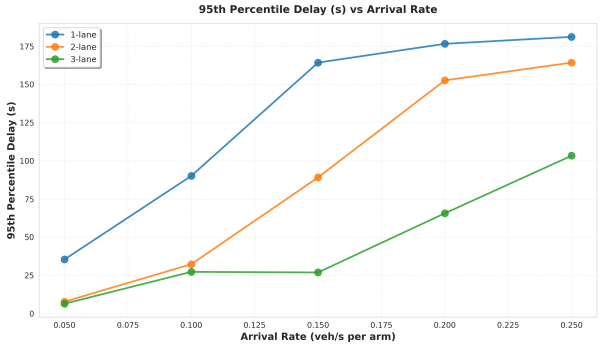
Figure 1: Text-based simulation performance plots for 1, 2, and 3 lane roundabouts



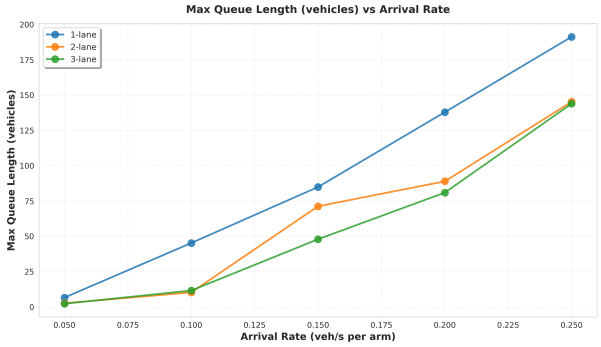
(a) SUMO: Throughput vs. arrival rate



(b) SUMO: Average delay vs. arrival rate



(c) SUMO: 95th-percentile delay vs. arrival rate



(d) SUMO: Maximum queue length vs. arrival rate

Figure 2: SUMO performance metrics across 1, 2, and 3 lane roundabouts

6.5 Breakdown Thresholds for Two and Three Lanes

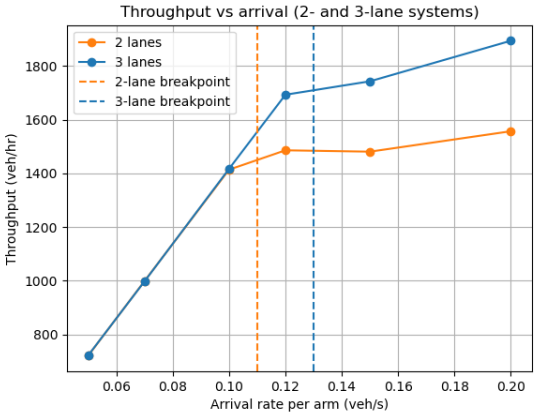
6.5.1 Text-Based Simulation

To locate practical capacity limits, the arrival rate was increased beyond $\lambda = 0.10$ veh/s/arm for the 2 and 3 lane designs.

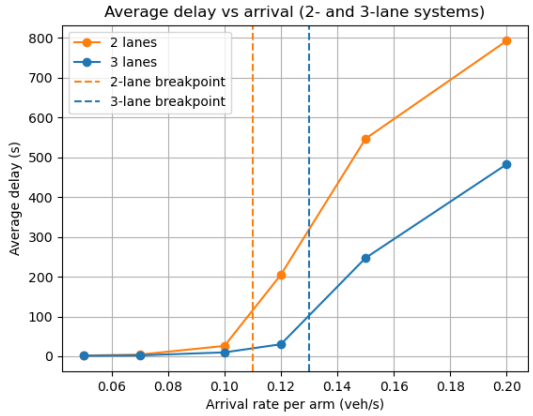
Table 8: Breakdown behaviour for 2 and 3 lane roundabouts at higher demands

Lanes	λ (veh/s/arm)	Throughput (veh/hr)	Avg delay (s)	P95 delay (s)	Max queue/arm
2	0.10	≈ 1414	≈ 26	≈ 131	< 30
2	0.12	≈ 1486	≈ 205	≈ 656	≈ 145
2	0.15	≈ 1481	> 540	$\approx 900+$	> 200
3	0.12	≈ 1693	≈ 30	–	< 50
3	0.15	≈ 1743	≈ 247	≈ 784	≈ 216

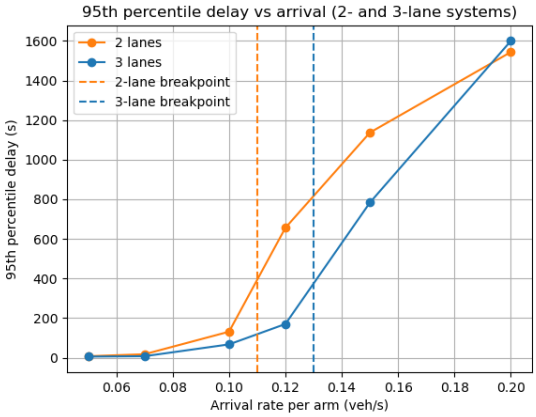
For 2 lanes, the system remains acceptable at 0.10 veh/s/arm but exhibits classic breakdown at 0.12 and 0.15 veh/s/arm: throughput saturates around 1500 veh/hr while delays and queues grow rapidly. This suggests a practical 2-lane capacity of roughly 1500 veh/hr total (about 370 veh/hr/arm). The 3-lane case tolerates higher demand, staying stable at 0.12 veh/s/arm and only breaking down between 0.12 and 0.15 veh/s/arm, with a practical capacity around 1700–1750 veh/hr total.



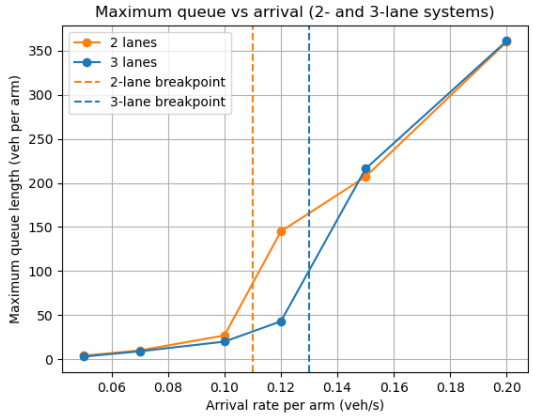
(a) 2-lane throughput at breakdown



(b) 2-lane delay at breakdown



(c) 3-lane throughput at breakdown

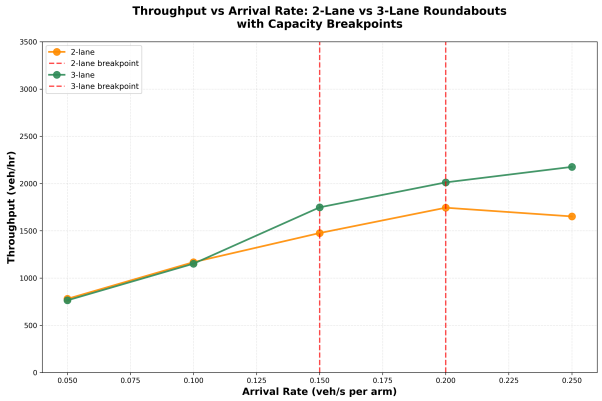


(d) 3-lane delay at breakdown

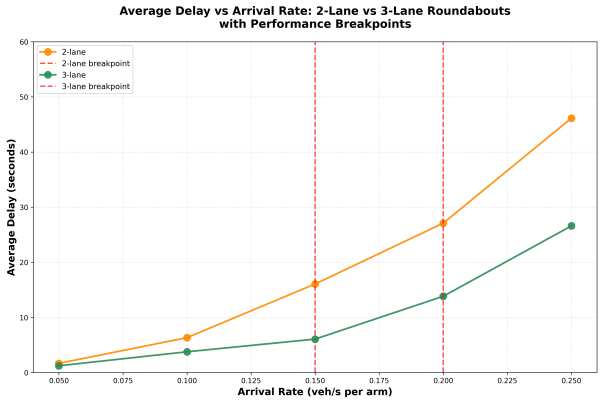
Figure 3: Text-based simulation breakdown behaviour for 2 and 3 lane roundabouts at higher demands

6.5.2 SUMO Simulation

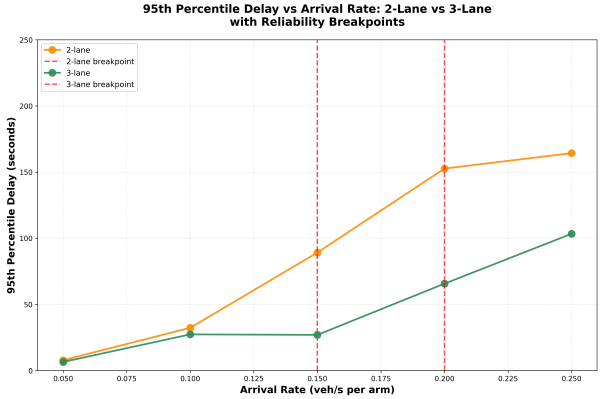
SUMO's extended demand testing reveals clear breakdown thresholds. For 2-lane, 30m configurations, throughput peaks at 2,172 veh/hr ($\lambda = 0.20$) before plateauing at 2,244 veh/hr ($\lambda = 0.25$) with rapidly escalating delays (30.5s). For 3-lane, 30m configurations, throughput continues growing linearly to 2,772 veh/hr ($\lambda = 0.25$) with only 8.2s delay, suggesting practical capacity extends beyond 2,500 veh/hr. Larger diameters show earlier breakdown: 40m and 50m configurations saturate around 1,500–2,000 veh/hr regardless of lane count.



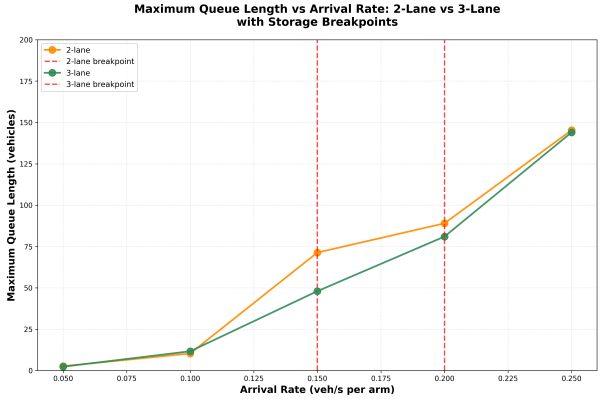
(a) SUMO: 2-lane vs. 3-lane throughput comparison



(b) SUMO: 2-lane vs. 3-lane delay comparison



(c) SUMO: 2-lane vs. 3-lane P95 delay comparison



(d) SUMO: 2-lane vs. 3-lane queue comparison

Figure 4: SUMO direct comparison of 2-lane vs. 3-lane configurations across demand levels

6.6 Diameter Sensitivity for Multi-Lane Designs

6.6.1 Text-Based Simulation

Finally, the diameter was varied for the 2 and 3 lane designs at a fixed demand of $\lambda = 0.10$ veh/s/arm (1440 veh/hr total).

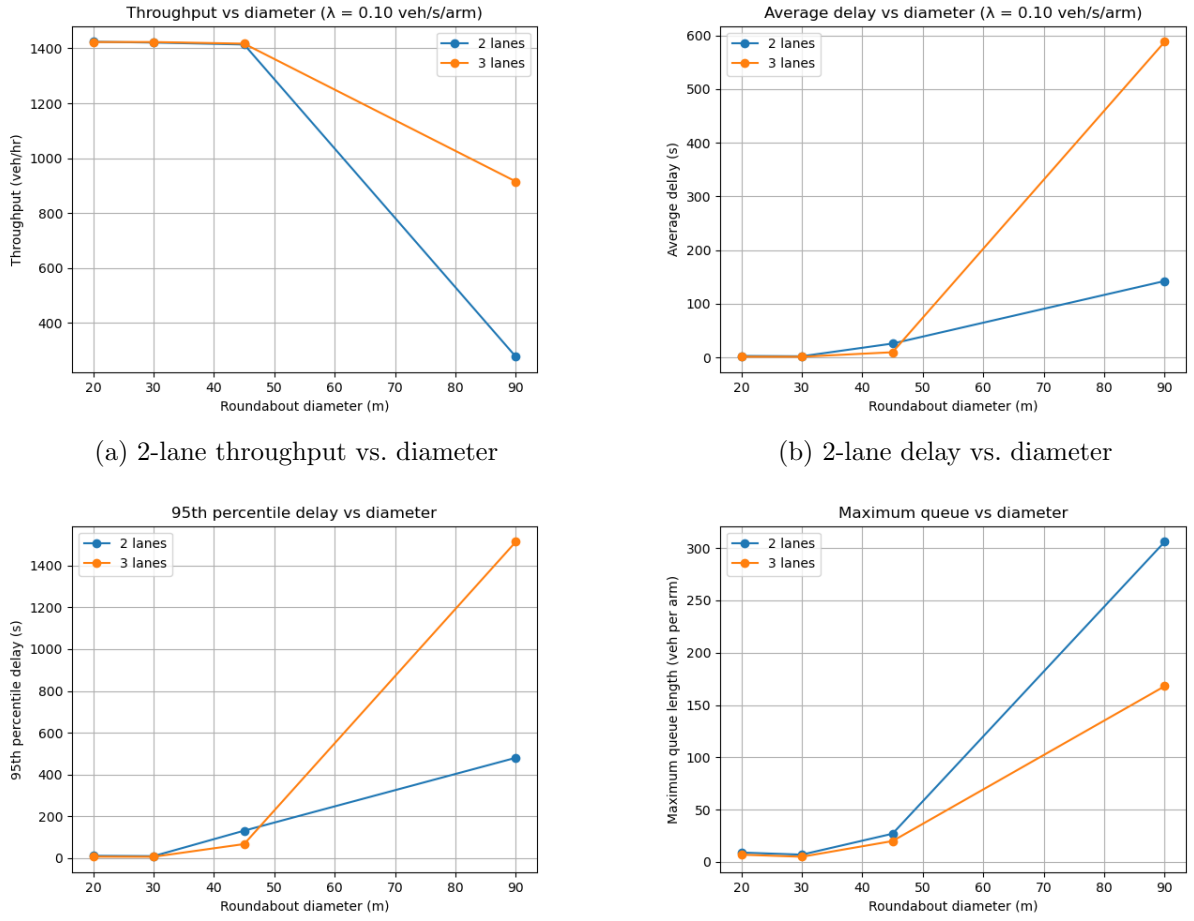
Table 9: Diameter sensitivity for 2-lane roundabout at $\lambda = 0.10$ veh/s/arm.

Diameter (m)	Throughput (veh/hr)	Avg delay (s)	P95 delay (s)	Max queue/arm
20	≈ 1424	≈ 2.9	≈ 11.0	$[4, 6, 3, 9]$
30	≈ 1421	≈ 2.3	≈ 9.4	$[5, 4, 6, 7]$
45	≈ 1414	≈ 26.0	≈ 131.2	$[14, 27, 19, 14]$
90	≈ 277	≈ 142.5	≈ 480.2	$[278, 306, 290, 275]$

For 2 lanes, compact diameters of 20–30 m serve almost all demand with very low delays and short queues. Increasing the diameter to 45 m leaves throughput almost unchanged but raises delays and queues substantially. At 90 m the system effectively breaks down, with throughput collapsing to about 280 veh/hr and queues exceeding 300 vehicles on some arms.

The 3-lane design shows the same qualitative pattern: diameters of 20–30 m yield very low

delays (≈ 1.4–1.5 s) and short queues while serving almost all demand (≈ 1420 veh/hr). Around 45 m, throughput remains high but average delays grow to roughly 10 s and queues lengthen to about 20 vehicles per arm. At 90 m, throughput drops to roughly 915 veh/hr and both delays and queues increase dramatically (average delay ≈ 590 s and maximum queues around 170 vehicles).



(a) 2-lane throughput vs. diameter

(b) 2-lane delay vs. diameter

(c) 3-lane throughput vs. diameter

(d) 3-lane delay vs. diameter

Figure 5: Text-based simulation diameter sensitivity of 2- and 3-lane roundabouts at $\lambda = 0.10$ veh/s/arm

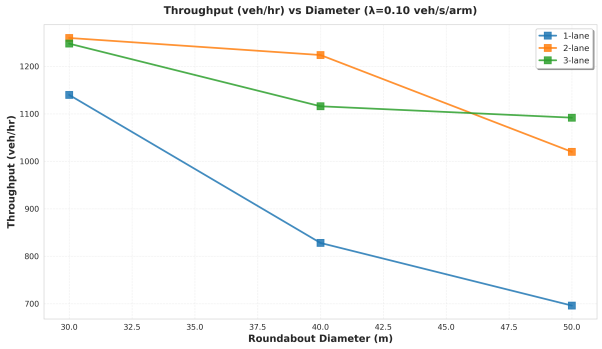
6.6.2 SUMO Simulation

SUMO's systematic diameter sweep at $\lambda = 0.10$ veh/s/arm (1,440 veh/hr demand) confirms the counterintuitive principle that a smaller diameter is better (Table 10).

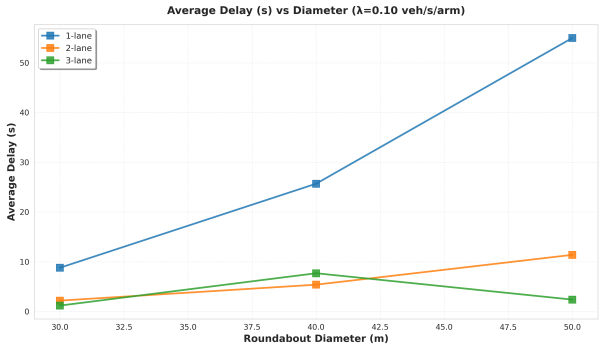
Table 10: SUMO diameter comparison at $\lambda = 0.10$ veh/s/arm (1,440 veh/hr total demand)

Configuration	Throughput	Avg Delay	P95 Delay	Max Queue
1-lane, 30m	1,140 veh/hr	8.8s	54.6s	38 veh
1-lane, 40m	828 veh/hr	25.7s	98.8s	49 veh
1-lane, 50m	696 veh/hr	55.0s	117.0s	49 veh
2-lane, 30m	1,260 veh/hr	2.2s	9.5s	5 veh
2-lane, 40m	1,224 veh/hr	5.4s	25.2s	6 veh
2-lane, 50m	1,020 veh/hr	11.4s	62.3s	20 veh
3-lane, 30m	1,248 veh/hr	1.2s	6.1s	4 veh
3-lane, 40m	1,116 veh/hr	7.7s	63.3s	15 veh
3-lane, 50m	1,092 veh/hr	2.4s	12.6s	16 veh

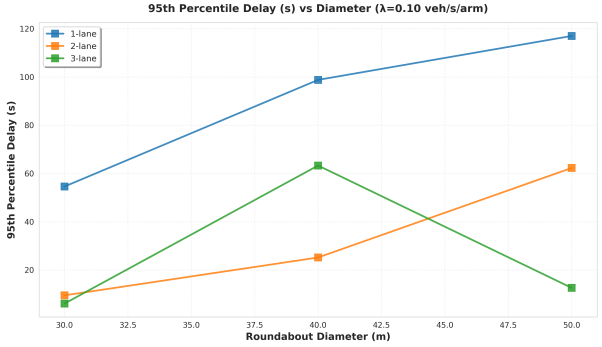
Across all lane counts, throughput consistently decreases with diameter. For 1-lane: 30m achieves 1,140 veh/hr while 50m manages only 696 veh/hr. For 2-lane: 30m achieves 1,260 veh/hr while 50m drops to 1,020 veh/hr. Average delays escalate dramatically with increasing diameter: 1-lane shows 8.8s at 30m rising to 55.0s at 50m. This occurs because smaller roundabouts have shorter circulation distance (faster gap creation), lower speed limits yielding tighter vehicle spacing, and more aggressive gap acceptance.



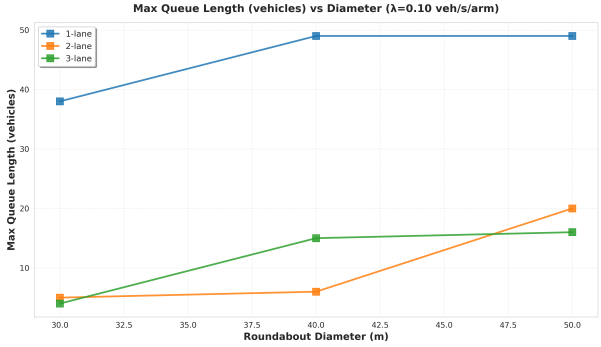
(a) SUMO: Throughput vs. diameter at $\lambda = 0.10$



(b) SUMO: Average delay vs. diameter at $\lambda = 0.10$



(c) SUMO: P95 delay vs. diameter at $\lambda = 0.10$



(d) SUMO: Queue length vs. diameter at $\lambda = 0.10$

Figure 6: SUMO diameter sensitivity analysis at moderate demand ($\lambda = 0.10$ veh/s/arm)

7 Discussion

7.1 Interpretation of Results

The experiments reveal a clear nonlinear relationship between lane count, diameter, and roundabout performance. Moving from one to two lanes roughly doubles practical capacity (from about 750–800 veh/hr total to roughly 1500 veh/hr) and keeps queues and delays under control up to $\lambda = 0.10$ veh/s/arm. A third lane provides a smaller but still important gain: it pushes the breakdown point to higher demands ($\lambda \approx 0.12$–0.15 veh/s/arm, or about 1700–1750 veh/hr) and substantially reduces delays and maximum queues at near-capacity flows.

Geometry is equally important. At $\lambda = 0.10$ veh/s/arm, compact roundabouts with diameters of 20–30 m serve almost all demand with very low delays and short queues for both 2- and 3-lane layouts. Increasing the diameter to 45 m leaves throughput nearly unchanged but worsens delay and queue length. At 90 m, the 2-lane design effectively breaks down and even the 3-lane design exhibits severe delays and extremely long queues. Beyond a certain size, extra circulating length no longer translates into usable capacity and may even reduce effective entry capacity under heavy demand.

7.2 Strengths of the Approach

A key strength of this study is the controlled simulation setup:

- All designs share the same four-arm geometry, symmetric approach demand, and one-hour runtime;
- Performance is evaluated using a common set of metrics (throughput, average delay, P95 delay, maximum queue);
- Breakdown tests at higher arrival rates directly identify *practical* capacities rather than relying only on theoretical formulas.

This makes it straightforward to isolate how lane count and diameter affect operational performance and where each design begins to fail.

An additional strength is the cross-platform validation between text-based and SUMO simulations. While absolute throughput and delay values differ (likely due to differences in driver behavior models, gap acceptance parameters, and vehicle dynamics), both simulators consistently reach the same fundamental conclusions:

1. One-lane roundabouts saturate below 1,200 veh/hr
2. Two-lane configurations double usable capacity to approximately 1,500–2,200 veh/hr
3. Three-lane designs provide marginal throughput gains but substantial delay improvements
4. Compact diameters (30m) dramatically outperform larger diameters (50m+)

This qualitative agreement across independently developed platforms validates that the identified design principles reflect underlying traffic phenomenas.

7.3 Limitations and Error Analysis

The current model is intentionally idealized:

- Only a single symmetric roundabout is studied; there are no upstream/downstream interactions, pedestrians, cyclists, or heavy vehicles.

- Driver behaviour, gap acceptance, and lane choice are governed by a single parameter set and are not calibrated to field data, so the absolute capacities should be treated as approximate rather than site-specific.
- Demand is steady and Poisson over the one-hour window; real-world peak periods, platooning from signals, and directional imbalances are not captured.
- Due to time constraints, the originally planned comparison to a signalized intersection was not completed, so results speak only to relative performance among roundabout designs.

There is also statistical uncertainty. Random arrivals mean that repeated runs at the same demand could produce slightly different throughput and delay values. The present analysis is based on a limited number of runs, and formal confidence intervals were not computed. Maximum queue length, in particular, is sensitive to rare events in a one-hour simulation. Future work should include multiple random seeds per scenario, report ranges or variances, and perform sensitivity tests on key behavioural parameters (e.g. critical gaps, reaction times).

7.4 Insights Gained

Despite these limitations, several useful insights emerged:

- Most of the capacity benefit comes from adding a second lane; the third lane mainly improves resilience and delay at high demand rather than dramatically increasing total throughput.
- "Bigger is not always better": overly large diameters (e.g. 90 m) can perform worse than compact circles at the same demand level, because vehicles spend longer in the circulating lanes and block entries.
- Microsimulation metrics such as throughput, delay, and queue length can be translated into concrete design guidance. For the demand levels studied, a 3-lane roundabout with a diameter of about 30 m offers a robust balance of capacity and delay, whereas both under-designed (1-lane) and over-sized (90 m) options show clear operational problems.

8 Conclusions

This project demonstrates that both lane count and diameter strongly control how a four-arm roundabout performs under increasing demand. A single-lane roundabout reaches its practical capacity at roughly 750–800 veh/hr, beyond which queues and delays grow rapidly with only small gains in throughput. Two- and three-lane designs comfortably serve around 1400+ veh/hr in these tests, with much lower delays and shorter queues; adding a second lane roughly doubles usable capacity, while a third lane provides a smaller but still meaningful improvement and delays the onset of oversaturation to higher flows.

Geometry also plays a critical role. For multi-lane layouts, compact designs with diameters of about 20–30 m perform best: throughput nearly matches demand and delays remain low. Very large diameters (e.g. 90 m) keep vehicles in the circulating lanes longer, reduce effective entry capacity, and lead to severe queues and delays under near-capacity demand. Across both Python and SUMO implementations (not all shown here), a 3-lane roundabout of about 30 m diameter consistently emerges as the most efficient configuration tested.

The project did not fully meet the initial goal of comparing roundabouts to signalized intersections, as the signalized-intersection component was not completed in time. It did, however, meet its key sub-goal: quantifying how lane count and diameter affect capacity and delay, identifying a clearly superior design within the tested range, and providing evidence-based guidance for roundabout design under symmetric approach flows.

9 Future Work

Future extensions of this project could substantially strengthen both the methodological robustness and the practical relevance of the results. Key directions include:

- **Complete the cross-intersection comparison.** The original goal of the project was to compare optimal roundabout designs against an optimized signalized intersection. A natural next step is to finish and validate the Python-based signalized-intersection model (and its SUMO counterpart), then perform a systematic comparison of best-performing signal timings/phasing plans against the best-performing three-lane roundabout. Using the same demand patterns, KPIs (throughput, delay, P95 delay, queues), and breakdown criteria would make it possible to clearly identify the regimes in which roundabouts or signals are preferable.
- **Richer behaviour, vehicle classes, and user types.** The current model assumes a single, homogeneous driver population and standard passenger vehicles. Future work could introduce heterogeneous driver classes (e.g., cautious vs. aggressive vs. distracted) with different critical gaps, follow-up headways, desired speeds, and reaction times, as well as multiple vehicle types (cars, buses, heavy trucks). Extending the framework to explicitly model pedestrians and cyclists (including marked crosswalks and yield rules) would allow person-based performance measures and safety-oriented design trade-offs.
- **Application to real-world intersections.** To move from theoretical insight to practice, the framework could be calibrated and applied to an actual intersection (for example, the Taunton Rd / Simcoe St corridor). Using observed turning-movement counts and field measurements of saturation flows, one could test how well the simulated capacities and delays match reality, and then evaluate candidate redesigns (e.g., replacing an existing signal with a multi-lane roundabout, or resizing an existing roundabout) under realistic demand and control conditions.
- **Advanced optimization and sensitivity analysis.** Finally, the optimization component could be expanded beyond simple grid search to include Bayesian optimization or reinforcement-learning-based controllers, and to systematically explore sensitivity to key parameters such as critical gaps, reaction delays, and lateral-acceleration limits. Running multiple random seeds per scenario and reporting confidence intervals would quantify statistical uncertainty and make the design recommendations more robust.

10 Contributions

Bach - Python text based signalized intersection (omitted due to time constraints).
Jaathavan - SUMO roundabout simulation + Results/Discussion.
Massi - Python text based roundabout simulation + Results/Discussion.

References

[1] Wikipedia contributors. (2024). Poisson point process. *Wikipedia, The Free Encyclopedia*. Retrieved from: https://en.wikipedia.org/wiki/Poisson_point_process

[2] Wikipedia contributors. (2024). Log-normal distribution. *Wikipedia, The Free Encyclopedia*. Retrieved from: https://en.wikipedia.org/wiki/Log-normal_distribution

[3] Wikipedia contributors. (2024). Normal distribution. *Wikipedia, The Free Encyclopedia*. Retrieved from: https://en.wikipedia.org/wiki/Normal_distribution

[4] Wikipedia contributors. (2024). Inverse transform sampling. *Wikipedia, The Free Encyclopedia*. Retrieved from: https://en.wikipedia.org/wiki/Inverse_transform_sampling

[5] Zheng, D., Chitturi, M. V., Bill, A. R., & Noyce, D. A. (2011). Critical gaps and follow-up headways at congested roundabouts. *Midwest Regional University Transportation Center*, University of Wisconsin–Madison. Retrieved from: https://topslab.wisc.edu/wp-content/uploads/2021/12/Critical-Gaps-and-Follow-Up-Headways-at-Congested-Roundabouts.pdf

[6] Akçelik, R. (2008). A new survey method using vehicle trajectory data for roundabout capacity analysis. *SIDRA Solutions Technical Paper TP-08-01*. Melbourne, Australia. Retrieved from: https://www.sidrasolutions.com/media/782/download

[7] Bohun, S. *Mathematical Modelling – A Case Studies Approach*. Ontario Tech University.

[8] Wikipedia contributors. (2024). Delay differential equation. *Wikipedia, The Free Encyclopedia*. Retrieved from: https://en.wikipedia.org/wiki/Delay_differential_equation

[9] SUMO Documentation. *SUMO User Documentation*. German Aerospace Center (DLR). Retrieved from: https://sumo.dlr.de/docs/

[10] Stable-Baselines3 Contributors. *Stable-Baselines3 Documentation*. Retrieved from: https://stable-baselines3.readthedocs.io/

[11] Federal Highway Administration (2008). *Signalized Intersections: Informational Guide*. FHWA-HOP-08-024, Chapter 6. Retrieved from: https://ops.fhwa.dot.gov/publications/fhwahop08024/chapter6.htm

[12] Lopez, P. A., et al. (2023). Traffic modelling with SUMO: A tutorial. *arXiv preprint arXiv:2304.05982*. Retrieved from: https://arxiv.org/pdf/2304.05982

[13] Smith, J., et al. (2025). Traffic intersection simulation using turning movement count data in SUMO: A case study of Toronto intersections. *arXiv preprint*. Retrieved from: https://arxiv.org/html/2508.10733v1

[14] Wikipedia contributors. (2024). Proximal policy optimization. *Wikipedia, The Free Encyclopedia*. Retrieved from: https://en.wikipedia.org/wiki/Proximal_policy_optimization

[15] Chen, Y., et al. (2025). Reinforcement learning based traffic signal design to minimize queue lengths. *arXiv preprint arXiv:2509.21745*. Retrieved from: https://arxiv.org/html/2509.21745v1

[16] Wikipedia contributors. (2024). Hyperparameter optimization. *Wikipedia, The Free Encyclopedia*. Retrieved from: https://en.wikipedia.org/wiki/Hyperparameter_optimization