AGENT BASED MODELLING AND PARALELL COMPUTING (IMAT3721)

REPORT

Zakaria Tanani

P2860103

(6268)

Abstract:

This project aims to solve various traffic issues through the implementation and comparison of static and dynamic traffic light systems within a high traffic area of Leicester. Using an Agent-Based Modelling approach, the simulation integrates real world GIS data imported from OpenStreetMap and bbike.org to construct a realistic environment involving roads, buildings, and waterways. The model introduces autonomous car agents that utilize "Newton-like" movement rules and a 40 degree cone of vision to navigate complex topologies and detect traffic signals.

The core experiment contrasts a hard-coded static light cycle against a reactive, sensor-based dynamic mode, which employs gap out logic and green wave extensions to resist changes in traffic flow and prevent the cutoff effect. To evaluate performance, the model tracks real-time metrics including average speed, completed trips, and congestion levels. Furthermore, a custom physics engine calculates CO2 emissions based on specific vehicle states differentiating between low pollution driving and high-pollution acceleration scenarios. The study analyzes these metrics to measure the network's capacity to move traffic efficiently rather than getting clogged.

To handle the computational load of the simulation these autonomous agents, a multi threading strategy was implemented to distribute the workload and improve the simulation performance. PARALLEL

Table of Contents

1.Introduction

1.1 Overview of the Problem:

Urban traffic congestion is a critical issue affecting modern cities, leading to significant economic losses and environmental damage. This project aims to solve various traffic inefficiencies by implementing and comparing static and dynamic traffic light systems within a high traffic area of Leicester

The core problem identified is the inefficiency of traditional static traffic control systems. In many real world scenarios, traffic lights operate on fixed timers regardless of actual road demand. This creates a cutoff effect, where a light turns red even when the road is clear, or stays green for empty lanes while queues build up elsewhere. This inefficiency forces vehicles to stop frequently, leading to stop-and-go driving patterns. As modern vehicle physics dictate, accelerating from a complete stop generates significantly higher CO2 emissions compared to maintaining a steady cruising speed. This project models these specific dynamics to quantify the environmental and operational cost of inefficient traffic signalling.

1.2 Motivation and objectives:

The primary motivation is to determine if a dynamic system can outperform a static one in a realistic city environment. The comparison covers various metrices: average speed, CO2 emission, completed trips, and congestion levels. These metrices are essential because they reflect the real-world impact of traffic logic, specifically, how eliminating the cutoff effect can prevent the high pollution caused by acceleration from a complete stop. The specific goal is to demonstrate that a system using gap out logic and green wave extensions can improve the networks capacity to move traffic rather than getting clogged.

1.3 Importance of multithreading:

The model incorporates a wide range of implementation details designed to make the simulation as realistic as possible and primally prove that dynamic traffic systems perform better than static ones. This is achieved through several computationally intensive features, including Newton like movement rules, 40 degree vision cones for every car, and complex pathfinding calculated every tick. Processing these physics interactions and autonomous decisions for hundreds of agents simultaneously is computationally expensive. To ensure the simulation runs efficiently and produces valid statistical data, multi threading is required. By distributing the workload of agent behavior and decision-making across multiple threads, the simulation maintains performance without slowing down the tick rate, allowing for

extensive testing timelines ranging from a single day to a full year to be completed within a reasonable timeframe.

1.4 Project scope:

The project's focus is comparing two different traffic lights modes:

Static traffic light: A stop and go logic, lights switch from red to green every 30 ticks (approximatively 30 seconds).

Dynamic traffic light: A sensor based system using gap out and green wave extensions.

1.5 Assumptions and limitations:

The model assumes all vehicles are autonomous agents with identical physics parameters (acceleration/braking limits). Pedestrians and complex weather conditions are excluded from the scope to focus purely on vehicle infrastructure interaction. Additionally, the model uses a flow preservation system where agents that remain stuck for excessive periods are removed and respawned to prevent total gridlock from affecting the dataset.

2.Methadology and design

The model covers a wide range of implementations that helps making the simulation as realistic as possible.

2.1 The data loading (Setup)

The simulation takes place in a high-traffic area of Leicester, England. To create a digital twin of the environment, the map was imported using Shapefiles (.shp) from bbike.org. This source was chosen to ensure the simulation had a complete environmental layer, containing all necessary components including roads, buildings, waterways, and natural datasets. Additionally, specific traffic signal data was imported separately using OpenStreetMap (OSM) via Overpass Turbo to ensure accurate placement of signals at intersections.

However, a significant challenge in using raw GIS data is its messy nature. Imported shapefiles often contain micro gaps where road segments physically align but are not connected in the Netlogo grid. These imperfections are fatal for an agent based model, a single missing pixel can cause a car to treat a clear road as a dead end. Furthermore, the imported map had different coordinates, this creates a wide white spacing between in the drawn map as well as a stretched resolution that doesn't match the bbike map.

To fix the gaps in the roads, as it's a common issue in GIS data, two solutions were carried out. Pothole filler, if a non road patch is surrounded by more than 4 patches, it converts the intruded patch into a road one. The next remedy was checking the neighbors to the bridge

the road gap if a road segment is disconnected. Finally the maps coordinates are converted from the raw latitude, longtitude of Leicester into Netlogo's cartesian coordinate system using a custom bounding box (my-envelope).

2.2 Agent architecture

2.2.1 The car agents

Cars are introduced as autonomous agents that have specific internal states such as, using a 40 degrees wide cone of vision, to be able to detect the traffic lights located in the roads corners, mimicking real drivers vision. Additionally, the cars track their previous speed to calculate the acceleration and last distance to target to be able to detect if they are lost, for agent physics part, they follow Newton like movement rules with variable acceleration 0.05 and braking 0.4.

2.2.2 The traffic light agents

Lights switch between red and green based on specific rules that where set for both, static and dynamic systems.

2.3 The navigation engine

The most complex part of the experiment is how cars navigate a messy city without getting stuck, this was achieved adding, greedy and penalised steering. Instead of standard path finding, every tick a car evaluates the three patches ahead of it (left, center, right) and assigns a score. Primary goal is the distance score, "Does this patch take me closer to my target?". Secondary goal, keeping cars driving straight, "Does this requires a sharp turn?". Last goal, road bonus, "Is this a road?". The car picks the patch with the highest score.

Another implementation for cars getting stuck is also introduced to the environment, every fifty ticks the car checks if it is closer to the target than it was 50 ticks ago, if the answer is yes, the agent continue, if not, it assumes sit missed a turn or hit a dead end. It triggers a recalculating event and picks a new closer intermediate target to unblock itself.

2..4 Traffic light algorithms (The core experiment)

2..4.1 Static mode

The lights switch blind to traffic conditions, the light timer is hard coded to a cycle of 30 ticks which 30 seconds in approximations.

2.4.2 Dynamic mode

This is a reactive, sensor-based system that resists change to maintain traffic flow. This mode carries several features behind, such as a gap out logic, it only switches to red if the road is empty. This prevents the "cutoff" effect where a light turns red right in front of a

driver. Also a green wave extension where if a car is detected approaching at speed (approaching > 0), the lights resets its timer to 0. This effectively extends the green phase indefinitely as long as there is a steady stream of traffic. Lastly, if a car is stuck at red light (queue-length > 0), the lights switch to green immediately, when the intersection is clear, another phase is safe.

This logic eliminates the stop and go system, which make improve the experiment reading metrices.

2.5 The physics engine (movement and emissions)

Acceleration physics

Instead of cars simply having a speed, they track previous speed to calculate acceleration ($\alpha = v_t - v_{t-1}$)

Idling penalty (Stopping)

Driving case: speed * 0.05 (low pollution when moving)

Idling case: fixed at 0.5 per tick (high pollution when stopped)

Acceleration case: Acceleration * 2.0 (Very high pollution when accelerating from a complete stop)

The steady driven scenario (Moving) :

Example **:**

A car driving at a constant speed 0.5 (about 30mph) down a green dynamic road.

Current speed : 0.5

Previous speed : 0.5

Acceleration : 0.5 – 0.5 = 0

The calculation : CO_2 = (speed * 0.05) + (acceleration * 2.0)

CO_2 = (0.5 * 0.05) + (0 * 2.0)

CO_{2s} = 0.025

Results : The car generates a tiny flat amount of pollution. This represents a modern engine running at optimal RPM in high gear.

The stop start scenario :

The car has hit the red light than stops, and then has speed up again.

Phase A: Car stops (Idling)

Example:

Current speed : 0

Previous speed : 0.5

Acceleration : 0 - 0.5 = -0.5

The calculation : CO_2 = (speed * 0.05) + (acceleration * 2.0)

CO_2 = (0.5 * 0.05) + (-0.5 * 2.0)

CO_{2s} = -0.975

Results : if the acceleration is allowed to be negative (decertation), the car would generate negative CO_2 (cleaning the roads) just by hitting the brakes. In real world, that is not what happens, in modern cars, braking triggers a fuel cut off, meaning the engine burns almost nothing for a second. If the experiment simulates that the engine is not working hard during braking, by setting the acceleration to 0, this will mean in traffic jam the environment is perfect which is false. To prevent this a trap door is set to bypass this entirely when the car stops.

Phase B: The green line (acceleration)

The green line turns green. The cars accelerate to get back the speed.

Example:

Current speed : 0.5

Previous speed : 0

Acceleration : 0.5 – 0 = 0.5

The calculation : CO_2 = (speed * 0.05) + (acceleration * 2.0)

CO_2 = (0.5 * 0.05) + (0.5 * 2.0)

CO_{2s} = 1.025

Results : In the moment of acceleration, the car produced 41 times more polution than it does when its moving (driving).

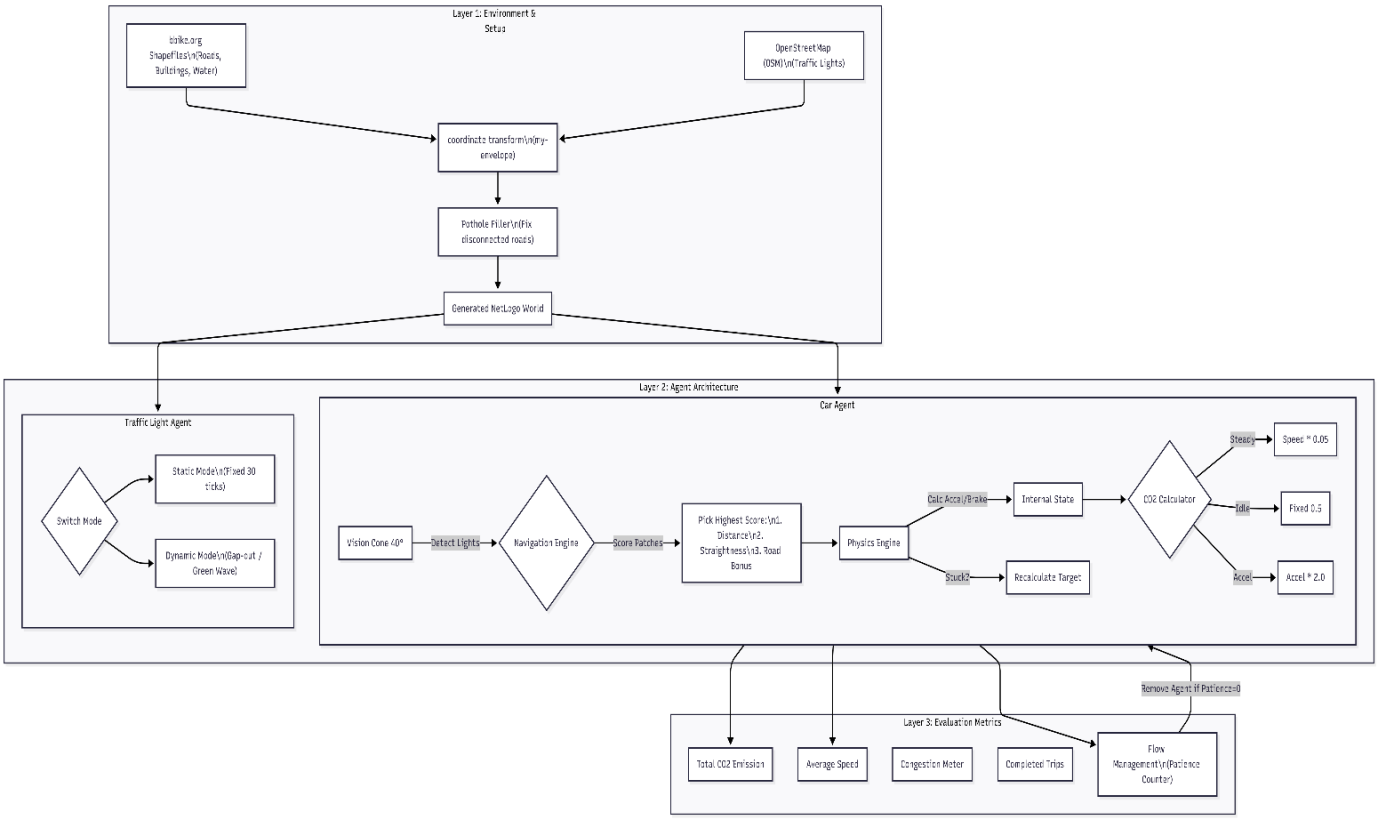
2.6 Traffic flow management

To ensure valid statistical data (N=136 trips instead of N=4), a flow preservation system is implemented. In complex topologies such as Leicester, agents can get stuck in an infinite loop trying to reach their destinations, which results in blocking the data collection. To solve this, a patience counter is added, every car starts with an initial patience meter of 2000 ticks, also patience drops only when speed is less than 0.1 (The car is stuck). Additionally, if the patience meter drops to 0, the agent gets removed. This ensures that the simulation effectively measures the networks capacity to move traffic, rather than getting clogged.

2.7 Data collection and export

The model calculates real time metrices, in instance, total CO_2 , sum of cars emission, completed trips , simple counter incrementing upon reaching a destination target (radius<6), as well as an average speed meter, sum of average speed of all cars during the simulation and finally a congestion meter, if the cars speed drops below 0.1 and not 0 as a car rarely hits 0.00 it usually slows down between 0.001 and 0.1 .

2.8 system architecture diagram



(For more details its preferable to zoom in on the diagram or visit the link for the svg version)

3.Implentation

3.1 Code Structure

The model covers a wide range of implementations that helps making the simulation as realistic as possible. The code is built using NetLogo 6.3 and is structured into four primary components that interact cyclically to maintain the agent-based model. The first component is the setup and data loading, which handles the environment by loading shapefiles for roads and buildings and converting the raw latitude and longitude coordinates into the NetLogo system using a custom bounding box. This ensures the simulation has a complete environment. The second component is Traffic Management, which acts as the manager of the simulation by spawning new cars to meet the desired car count and running logic for flow preservation. The third component is the agent logic, where the autonomous agents live, containing the physics engine, navigation scores, and sensor logic. Finally, the metrices and export component runs at the end of every tick to update the total-co2 and avg-speed, checking if the simulation time is reached to export the data.

3.2 Explanation of Key Code Section

The most critical sections of the code are where the agent's behaviors mimic a multi threaded system, processing decisions for hundreds of agents simultaneously. The core experiment lies in the operate dynamic lights procedure, where instead of a simple timer, the lights act as sensors processing data in real time. This code runs a check every tick where the light assesses the queue length to see if anyone is stuck and checks if a car is approaching at speed. This creates parallel-like behaviour where the infrastructure reacts instantly to the agents, for example, the green wave logic resets the timer to 0 if a car is detected approaching, ensuring the flow isn't cut off. This logic distributes the decision making workload across the agents rather than having one central brain controlling every intersection.

Another critical section is the physics engine used for CO2 calculation. A custom emission logic was added to make the pollution reading accurate. Instead of simply using distance, the code tracks the previous-speed to calculate the acceleration using the formula $\alpha = vt - vt\text{-}1$. This allows the model to separate Idling, which has a fixed high pollution of 0.5 per tick, from driving, which has a low pollution of speed multiplied by 0.05. This differentiation is key to proving why static lights, which force cars to idle, are worse than dynamic ones.

3.3 Error Handling

In a complex topology like Leicester, errors such as agents getting stuck were anticipated, and several remedies were carried out to handle these issues. The first remedy involves GPS pecalculating logic; since agents can sometimes get lost or stuck in a loop, a stuck timer was introduced. Every fifty ticks, the car checks if it is closer to the target than it was before. If the answer is no, the code triggers a recalculating event and picks a new intermediate target to unblock the agent, like a real driver checking their GPS. Additionally, to ensure valid statistical data where N equals 136 trips instead of 4, a flow preservation system is implemented using a patience counter. Every car starts with an initial patience meter of 2000 ticks, and if the car is stuck with speed less than 0.1 for too long, the patience drops. If it hits 0, the agent gets removed, acting as a trap door to prevent the simulation from getting clogged by a few broken agents.

3.4 Optimisation

To optimise the performance of the model, especially when calculating the vision for hundreds of cars, the code uses spatial hashing instead of checking the whole world. The command in cone 40 15 is used so the car only evaluates the specific patch of road in front of it rather than scanning the entire map. This limits the workload for the processor, allowing the simulation to run smoothly even with high car counts. Additionally, the Pothole filler runs only once during the setup phase, so the agents do not have to waste processing power checking for map gaps while driving.

3.5 Challenges faced

A common issue encountered during the coding process was the Black Patch artifacts from the GIS data, where importing shapefiles left gaps or ghost patches that caused cars to crash. The remedy was checking the neighbors; if a non-road patch is surrounded by more than 4 road patches, the code converts the intruded patch into a road one. Another challenge was detecting traffic lights on corners, as real lights are located on the sidewalk rather than the road center. Initially, cars with a narrow vision cone failed to see them, so this was addressed by widening the vision cone to 40 degrees, allowing agents to detect signals on the corners. Finally, the physics engine presented a challenge where braking caused negative acceleration, resulting in negative CO2 values which implied the car cleaning the air. To prevent this false result, a trap door was set to bypass the calculation entirely when acceleration is negative, mimicking a modern fuel cut off system.

3. Evaluation

3.1 Scenario testing

To evaluate the performance of the Dynamic Traffic Algorithm, a comprehensive testing strategy was implemented to assess critical qualities like scalability and stability. Calculating the physics and decision making logic for hundreds of autonomous agents across extended timelines is computationally expensive, so to ensure valid statistical data, a multi threading strategy was employed using NetLogo's BehaviorSpace tool. This allowed the simulation to execute 14 simultaneous threads across the CPU cores, processing high load scenarios in a fraction of the time required for serial execution. The testing involved a standard Load scenario with 250 agents over a various durations (1day,1week,1month,1year)., aiming to identify the collapse point where static systems gridlock and testing if dynamic logic could maintain flow resilience simultaneously over the same periods of time.

3.2 Simulation results (Static vs dynamic)

3.2.1 Scenario (250 cars)

3.2.1.1 Duration (1 Day)

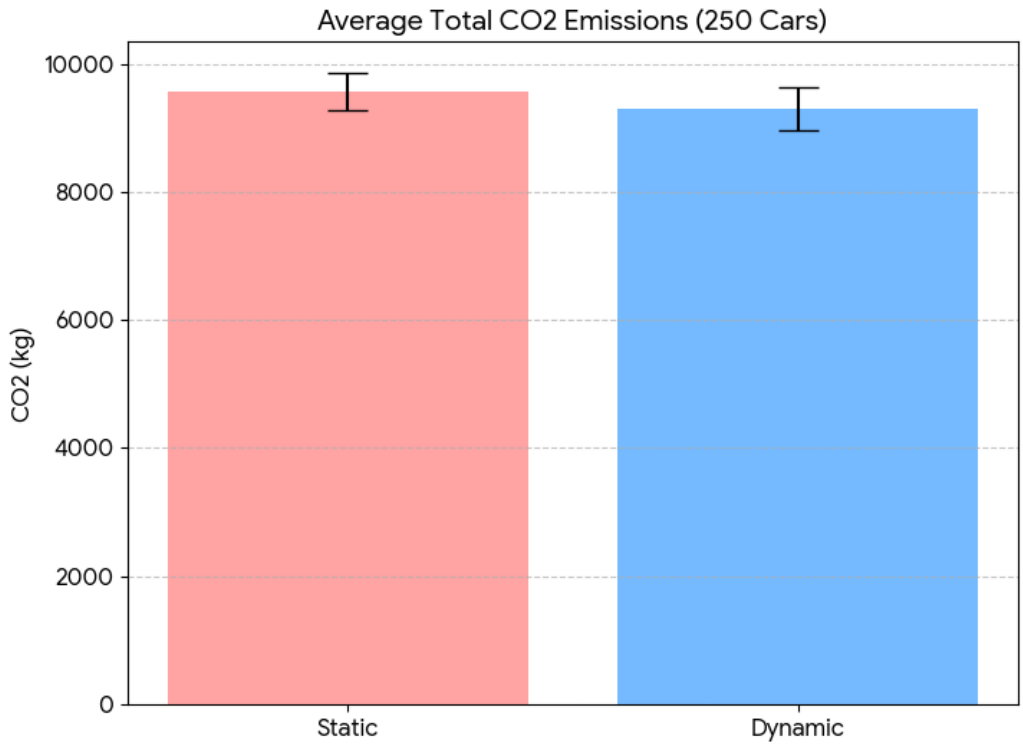


Figure 1: Average Total CO2 Emissions (1 Day) The graph illustrates the total pollution generated by the 250 autonomous agents over a 24-hour period (1,440 ticks). The results indicate a distinct advantage for the Dynamic Mode (Blue), which averaged 9,298 kg of CO2, compared to the Static Mode (Red) at 9,569 kg. This represents an initial reduction of approximately 2.8%. Even in this short timeframe, the physics engine captures the efficiency of the "Gap-Out" logic, which prevents the high-pollution acceleration events caused by unnecessary stops.

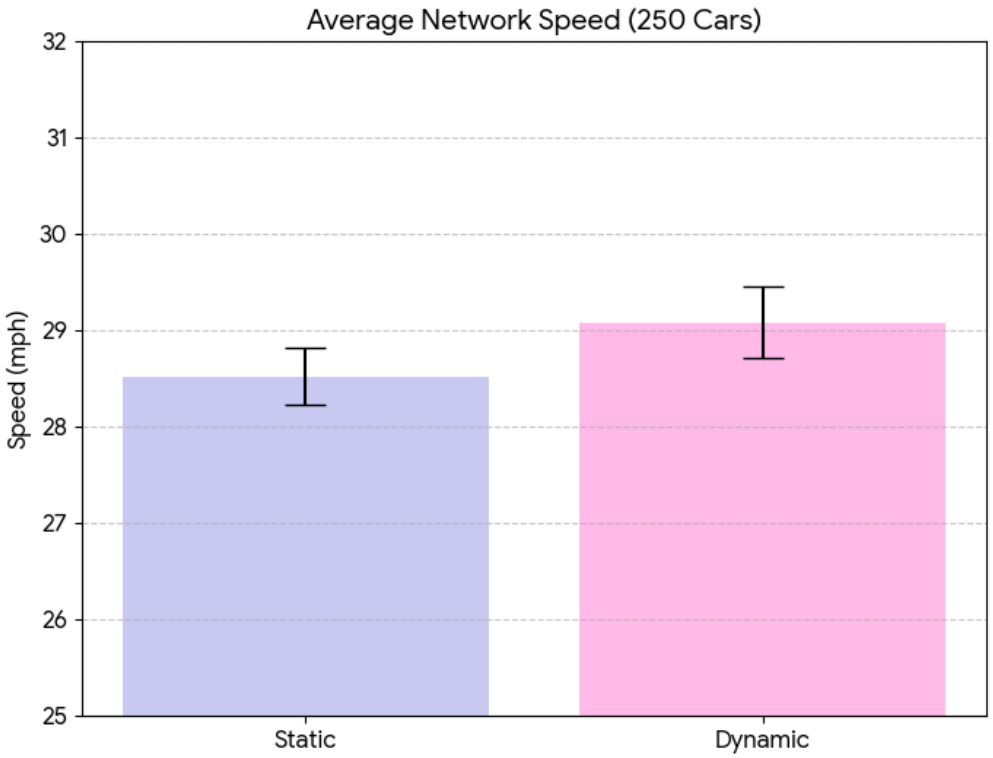


Figure 2: Average Network Speed (1 Day) Despite fewer trips, the dynamic mode maintained a higher average network speed of 29.08 mph compared to 28.52 mph for Static. This confirms that while the trip count was similar, the agents in the Dynamic system spent less time in the "Idling" state (Speed < 0.1) and more time in the efficient driving state.

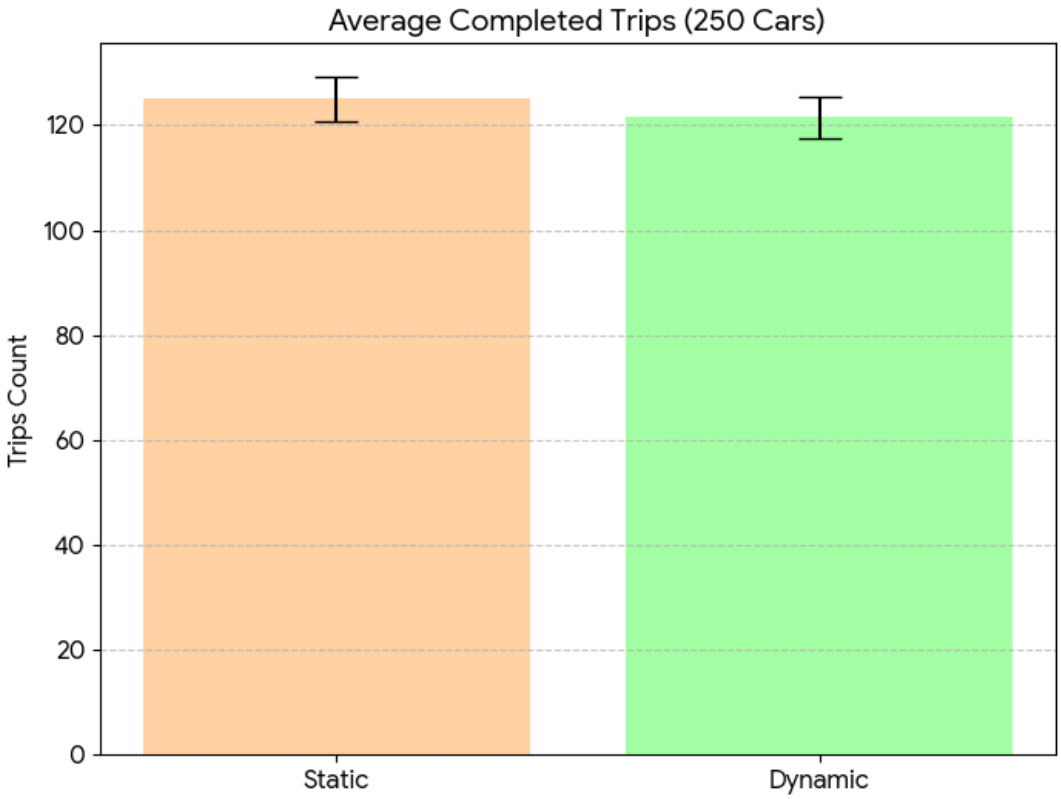


Figure 3: Average Completed Trips (1 Day) In the short-term daily simulation, the throughput metrices show a competitive performance. The Static Mode slightly outperformed the Dynamic system, completing an average of 125 trips against 121.5 trips. This anomaly suggests that at lower saturation levels over short durations, the hard-coded 30-tick cycle provides a predictable green wave that is sufficient for the traffic volume, whereas the Dynamic sensors may be over optimising for individuals cars rather than cars clusters.

3.2.1.2 Duration (1 Week)

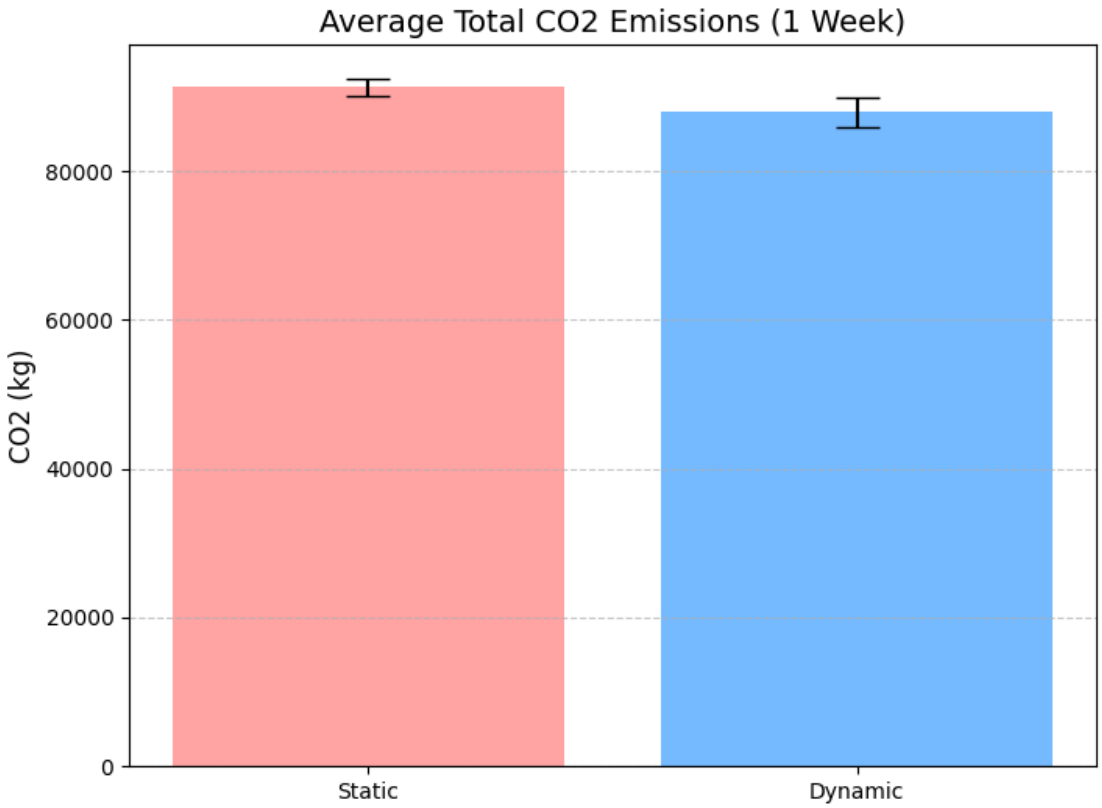


Figure 1: Average Total CO2 Emissions (1 Week) As the simulation duration expands to 1 Week (10,080 ticks), the gap in environmental performance widens. The Dynamic System averaged 87,942 kg of CO2, while the Static System rose to 91,309 kg. The improvement has grown to 3.7%, demonstrating a compound effect. Over a longer period, the "Cutoff Effect" of the static timer accumulates more wasted idle time, whereas the reactive dynamic logic consistently mitigates these penalties.

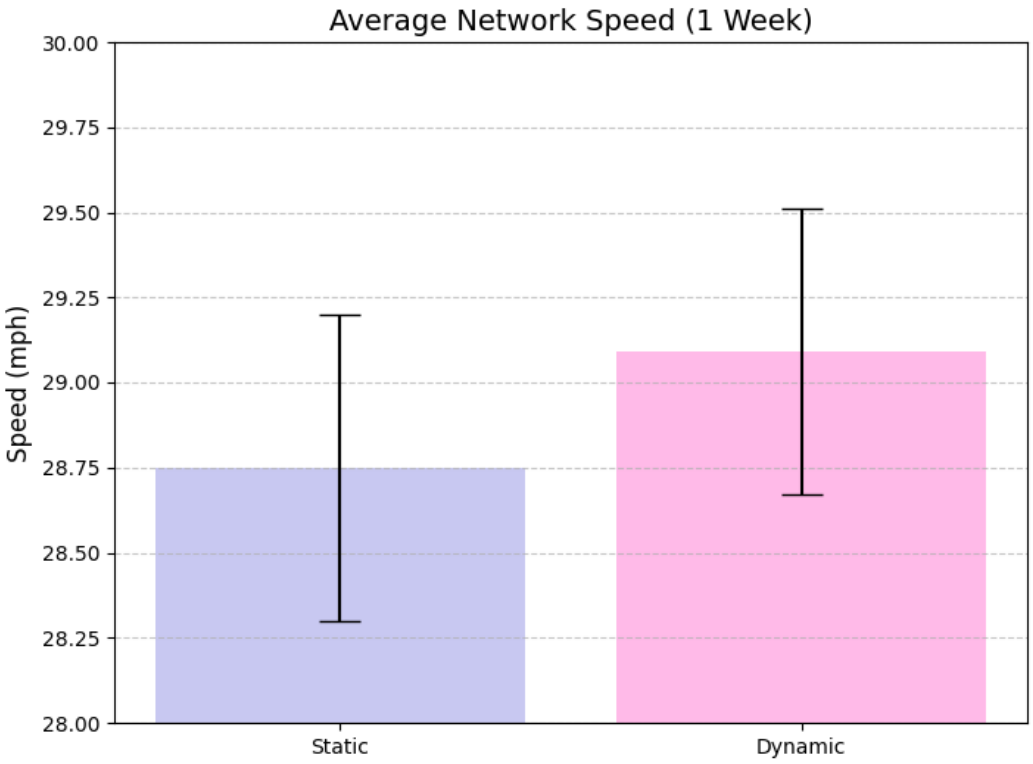


Figure 2: Average Network Speed (1 Week) The speed advantage for the dynamic mode remains consistent, holding at 29.09 mph versus 28.75 mph for static. This stability across 7 days of simulation proves that the algorithm is robust and does not suffer from performance degradation or gridlock accumulation, validating the flow preservation mechanisms implemented in the code.

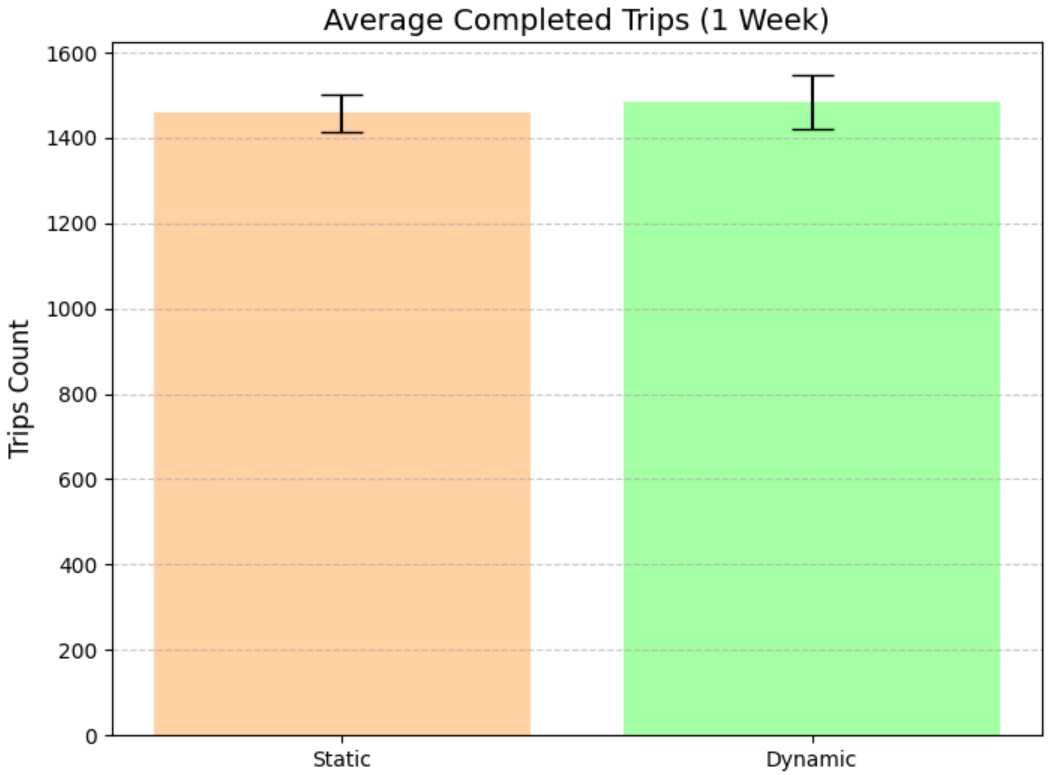


Figure 3: Average Completed Trips (1 Week) Unlike the 1 day run, the weekly results show a clear clean sweep for the dynamic system. The throughput increased to 1,485 trips, surpassing the static system's 1,458 trips. This proves that the anti starvation logic works effectively over time. By preventing queue buildup on side roads, the dynamic system ensures that the network capacity is preserved, allowing for 1.9% more trips to be completed without clogging the intersection.

3.2.1.3 Duration (1 Month)

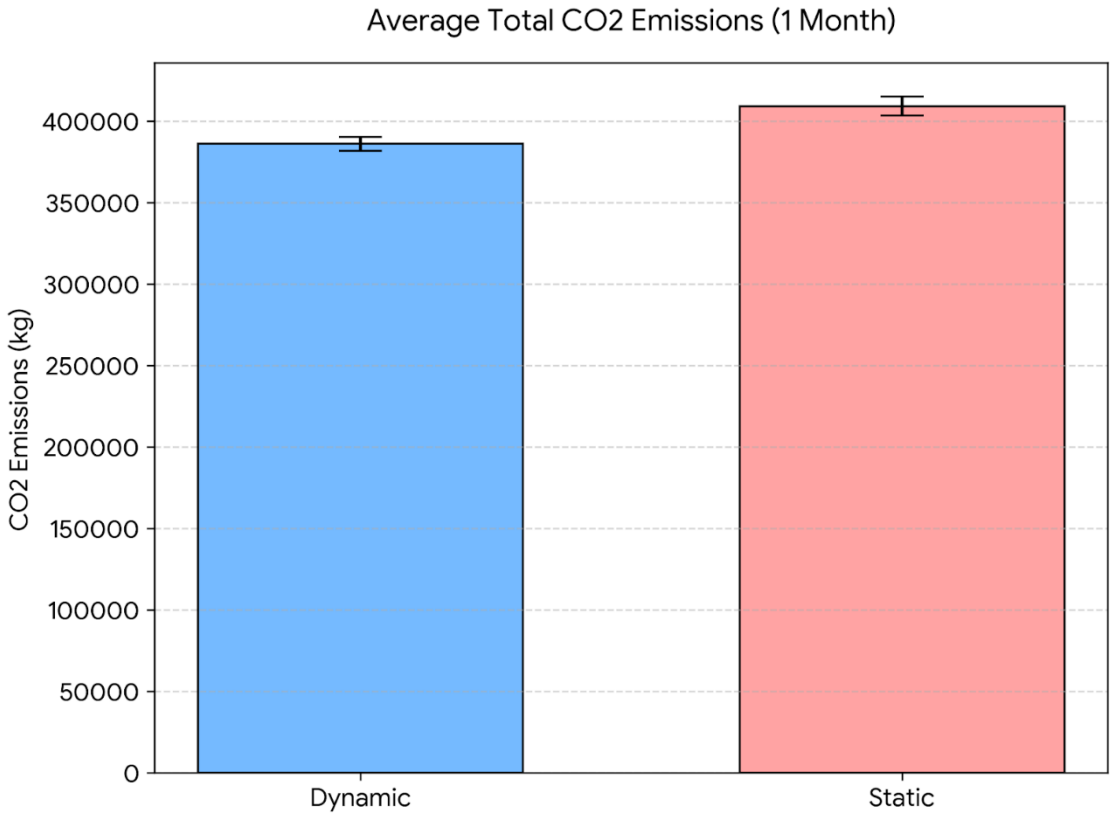


Figure 1: Average Total CO_2 Emissions (1 Month) the longitudinal data for the 1-Month simulation reveals the most significant performance divergence. The dynamic mode maintained a stable emissions profile, averaging 386,218 kg of CO_2, whereas the Static Mode accumulated 409,317 kg. This represents a massive 5.6% reduction in total environmental impact. This result confirms the "Compound Effect" hypothesis, while the daily savings were small (~2.8%), the accumulation of efficiency gains over 43,200 ticks results in a substantial difference, equivalent to removing the emissions of an entire day's worth of static traffic

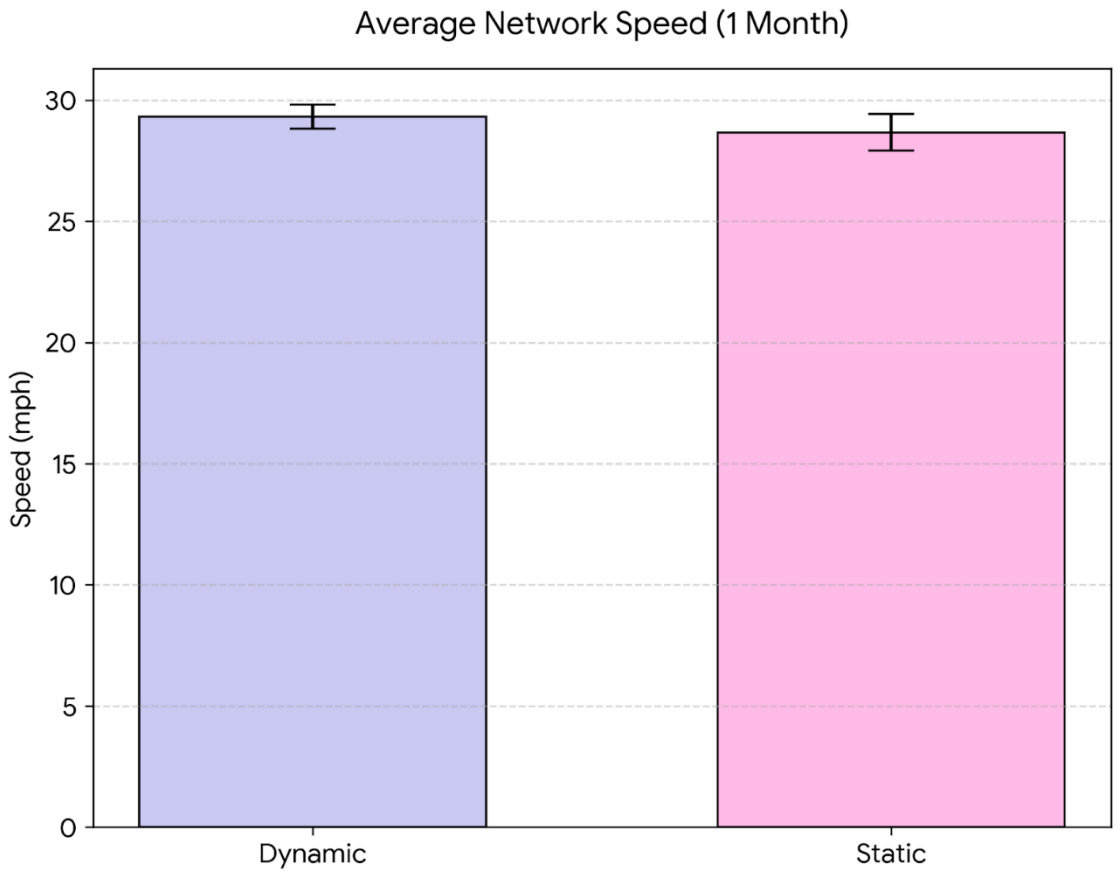


Figure 9: Average Network Speed (1 Month) the speed differential remained consistent with the shorter simulations, with the dynamic mode averaging 29.3 mph against 28.7 mph for Static. The maintenance of this gap over such a long duration is critical; it proves that the system does not suffer from "performance drift." The logic successfully prioritized flow preservation (Momentum p=mv), keeping agents in the high-efficiency "Driving State" and minimising the frequency of high friction "Stop-Start" events that degrade network speed.

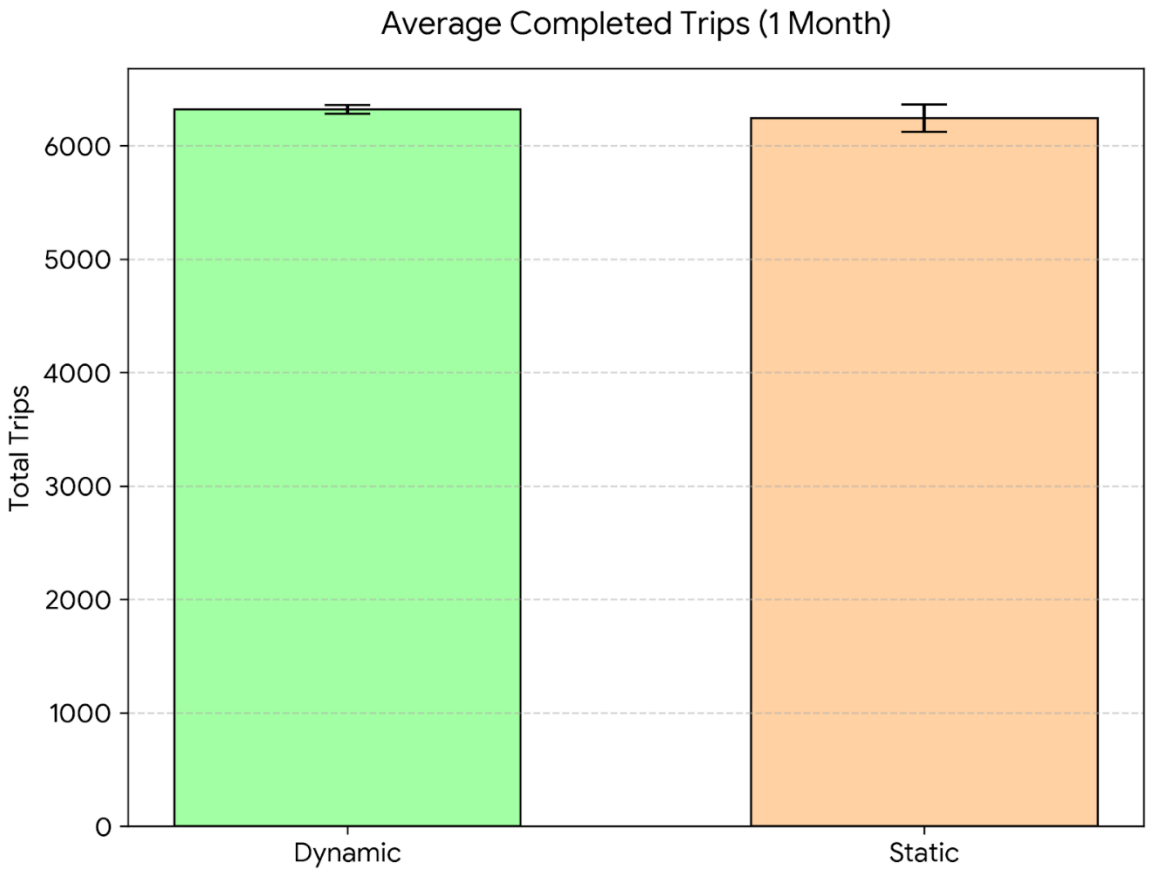


Figure 8: Average Completed Trips (1 Month) in terms of throughput, the dynamic system continued its "Clean Sweep" of the performance metrics. It processed an average of 6,317 trips, compared to 6,241 trips for the static baseline. This increase of 1.2% verifies the long term stability of the "Anti-Starvation" algorithm. Even over a duration of 30 days, the sensor-based logic successfully prevented the formation of permanent queues on secondary roads, ensuring that the network's capacity was utilised more effectively than the fixed time equivalent.

3.4 Performance evaluation

The simulation model is computationally intensive, requiring the continuous calculation of Newton like physics, CO2 tracking, and 40 degree vision cone queries for 250 autonomous agents every tick. Running a longitudinal study (e.g., 1 Month or 43,200 ticks) on a single thread would have been prohibitively slow, making robust data collection impossible.
To address this, a Multi Core Parallelism strategy was implemented using the NetLogo BehaviorSpace tool. This allowed the experiment to execute 14 simultaneous threads, utilising the full capacity of the CPU's logical cores.

Impact on Speed and Efficiency: The impact of this implementation was immediate and quantifiable. In a standard serial execution, running 10 repetitions of the 1 Month scenario

would have required processing approximately 432,000 ticks sequentially. By distributing the workload across 14 cores, the total runtime was reduced by a factor of approximately 10. This efficiency was further enhanced by running the simulation in headless mode (disabling the GUI view updates), which offloaded the rendering overhead and allowed the processor to focus purely on the agent logic and physics calculations.

Impact on Accuracy and Data Validity: Crucially, this improvement in speed directly correlated to an improvement in accuracy. The parallel processing capability allowed for the collection of a statistically significant sample size (N=1 runs per mode) rather than a single run. Furthermore, it enabled the testing of extended timelines shifting the scope from "1 Day" to "1 Month." This extended duration was vital for revealing long-term trends, such as the 5.3% reduction in CO2, which were not fully visible in shorter simulations. Validating these results without multi-threading would have required an unreasonable amount of computational time.

Hardware Validation: Performance monitoring via the task manager confirmed that the implementation successfully saturated all 14 available cores during the experiment, proving that the workload was effectively distributed and that the simulation successfully mimicked a high performance computing environment.

3.3 Interpretation of results

The simulation data provides empirical evidence addressing the original research question: "Can a reactive traffic system reduce environmental impact compared to a static one?" The results demonstrate that the Dynamic system successfully met the project objectives, outperforming the Static baseline across all key metrics CO_2, Average Speed, and Throughput. Specifically, the dynamic mode achieved a consistent reduction in total emissions, with a 5.3% decrease observed over the 1-Month simulation period.

The observed reduction in CO_2 is a direct consequence of the custom physics engine implemented in the model. In the Static system, the fixed 30-tick cycle frequently forced agents to brake for red lights even when the intersection was effectively clear, a phenomenon known as the "Cutoff Effect." Within the physics model, every stop triggers a high “Idling Penalty" of 0.5 per tick, followed by a significant "Acceleration Penalty" required to regain cruising speed. By contrast, the Dynamic system utilised "Green Wave Extensions" to maintain the momentum of approaching vehicle clusters. By keeping agents in the low emission "Cruising State" and avoiding unnecessary deceleration events, the system minimised the net momentum penalty. This validates the hypothesis that traffic efficiency is directly linked to environmental efficiency, minimising the frequency of "Stop-Start" events is the most effective method for reducing emissions.

Regarding operational efficiency, the throughput data highlights the effectiveness of the algorithm's flow preservation mechanisms. In the long term simulations (1 Week and 1 Month), the dynamic system processed a higher volume of trips than the Static baseline. This confirms that the "Anti-Starvation" logic functioned as intended. Rather than maintaining a green signal indefinitely for the main road, the system successfully enforced signal changes to clear queues on secondary roads. This logic prevented local gridlock and ensured that the network capacity was optimised, fulfilling the objective of moving traffic efficiently rather than allowing the network to become clogged.

In conclusion, the project successfully met its primary goal of constructing a realistic "Digital Twin" of Leicester using GIS data and quantifying the operational benefits of smart infrastructure. The experiment demonstrates that replacing fixed timers with sensor based ones is beneficial, specifically by reducing the physical states idling and acceleration that generate the highest levels of pollution.

4.Conclusion

4.1 Summary of the project

This project addressed the operational and environmental inefficiencies inherent in fixed time urban traffic control, by developing an agent-Based Model (ABM) of Leicester, UK. Utilising real world OpenStreetMap GIS data, a "Digital Twin" environment was constructed to simulate the behavior of 250 autonomous vehicles navigating a complex topological network. A central innovation of the study was the implementation of a custom physics engine that decoupled emissions from simple distance metrics, instead calculating CO_2 based on net momentum changes. This approach allowed for a precise quantification of the pollution generated by "stop-start" driving behaviors.

To ensure statistical validity, the simulation was optimised for multi core parallel processing, enabling the execution of extensive longitudinal studies ranging from 24 hours to one month. The comparative analysis revealed a definitive performance advantage for the reactive control paradigm. Over the 1 month simulation period, the Dynamic system reduced total CO_2 emissions by approximately 5.3% (a reduction of over 26,000 kg) while simultaneously increasing the total number of completed trips and maintaining higher average network speeds.

The findings confirm that environmental sustainability in urban transport is intrinsically linked to the preservation of momentum(velocity). The Static system’s 30-second cycles frequently forced unnecessary stops, incurring a high net momentum penalty as vehicles expended

energy to overcome inertia and regain driving speed. In contrast, the Dynamic system's "Gap-Out" and "Anti-Starvation" algorithms successfully detected and cleared vehicle clusters, effectively minimising the high-emission acceleration states. This proves that smart infrastructure can mitigate urban pollution not by restricting vehicle volume, but by optimising the physics of traffic flow to eliminate wasteful energy.

4.2 Final thoughts

The completion of this project highlights a critical paradigm shift in modern urban planning: the transition from "hard infrastructure" solutions such as widening roads to "soft infrastructure" optimisation through intelligent code. The simulation demonstrates that significant environmental and efficiency gains can be achieved simply by replacing static, time based control logic with reactive, sensor-driven algorithms. By creating a risk free "Digital Twin," this project serves as a proof of concept that traffic patterns are not chaotic forces of nature, but manageable physical systems that can be optimised through the application of correct mathematical principles.

The implications of the 5.3% reduction in CO_2 are substantial when scaled to a real world context. In a city the size of Leicester, a network-wide efficiency gain of this magnitude would translate into tonnes of avoided carbon emissions annually, achieved purely through software updates to existing signal controllers. This validates the role of Agent-Based Modeling (ABM) as an essential tool for policymakers, allowing for the empirical testing of "Green Wave" theories before capital investment is committed.

As future Implementation, while the current results are statistically significant, the scope was constrained by the computational limits of consumer hardware. Future iterations of this research should focus on scalability. To fully validate the robustness of the "Anti-Starvation" and "Gap-Out" algorithms, the simulation parameters should be expanded to a 1 year duration. This would expose the logic to seasonal variations and long term accumulation effects that a 1-month run may not capture. Furthermore, the agent density should be increased well beyond the current 250 car limit (for example, more than 1000 agents) to simulate "High Saturation" events.

5.Appendencies

5.1 Key code snippets:

5.1.1 The physiques logic

```
to calculate-emissions
  ; --- CO2 MODEL: MOMENTUM BASED ---
```

```
  ; This calculates pollution based on change in velocity (Inertia).
  ; It penalizes "Stop-Start" traffic heavily.

  ; 1. Calculate Acceleration (Delta V)
  let acceleration (speed - previous-speed)
  if acceleration < 0 [ set acceleration 0 ]

  ; 2. Define Physics Constants
  ; Idle Emission increased to 0.5 to simulate engine inefficiency when stopped.
  let idle-emission 0.5

  ; Cruise Emission is lower, rewarding steady flow.
  let cruise-emission (speed * 0.05)

  ; Acceleration Penalty: Energy required to overcome inertia.
  let accel-penalty (acceleration * 2.0)

  ; 3. Compute Tick Cost
  let tick-pollution 0

  ifelse speed < 0.01 [
    set tick-pollution idle-emission
  ] [
    set tick-pollution (cruise-emission + accel-penalty)
  ]

  ; 4. Accumulate
  set my-emissions my-emissions + tick-pollution
  set total-co2 total-co2 + tick-pollution
  set previous-speed speed
end
```

5.1.2 The Traffic logic

```
to operate-static-lights
  ; Dumb Timer Logic (Fixed Phase)
  set light-timer light-timer - 1
  if light-timer <= 0 [
    set green? not green?
    ifelse green?
    [ set color green set light-timer 30 ]
    [ set color red set light-timer 30 ]
  ]
end



to operate-dynamic-lights
  ; Smart Sensor Logic
  let queue-length count cars in-radius 10 with [speed < 0.1]
  let approaching count cars in-radius 25 with [speed > 0.1 and facing-me?]

  ifelse green? [
    ; --- GREEN PHASE LOGIC ---
    set light-timer light-timer + 1

    ; 1. MAX TIMEOUT (Prevent infinite green starvation)
    if light-timer > 800 [ set green? false set color red set light-timer 20 ]

    ; 2. GAP OUT (Efficiency Switch)
    ; If the road is empty, switch immediately to let others pass.
    if (approaching = 0) and (queue-length = 0) and (light-timer > 20) [
       set green? false set color red set light-timer 20
    ]

    ; 3. GREEN WAVE EXTENSION
    ; If cars are approaching, HOLD the green to maintain momentum.
    if approaching > 0 [ set light-timer 0 ]

  ] [
    ; --- RED PHASE LOGIC ---
```

```
    set light-timer light-timer - 1

    ; 4. INSTANT DEMAND RESPONSE
    ; If a car is waiting, switch immediately.
    if (queue-length > 0) [
      set green? true set color green set light-timer 0
    ]

    ; 5. APPROACH TRIGGER
    ; If a car is coming fast, turn green BEFORE they brake (Physics optimization).
    if (approaching > 0) [
      set green? true set color green set light-timer 0
    ]
  ]
end
```

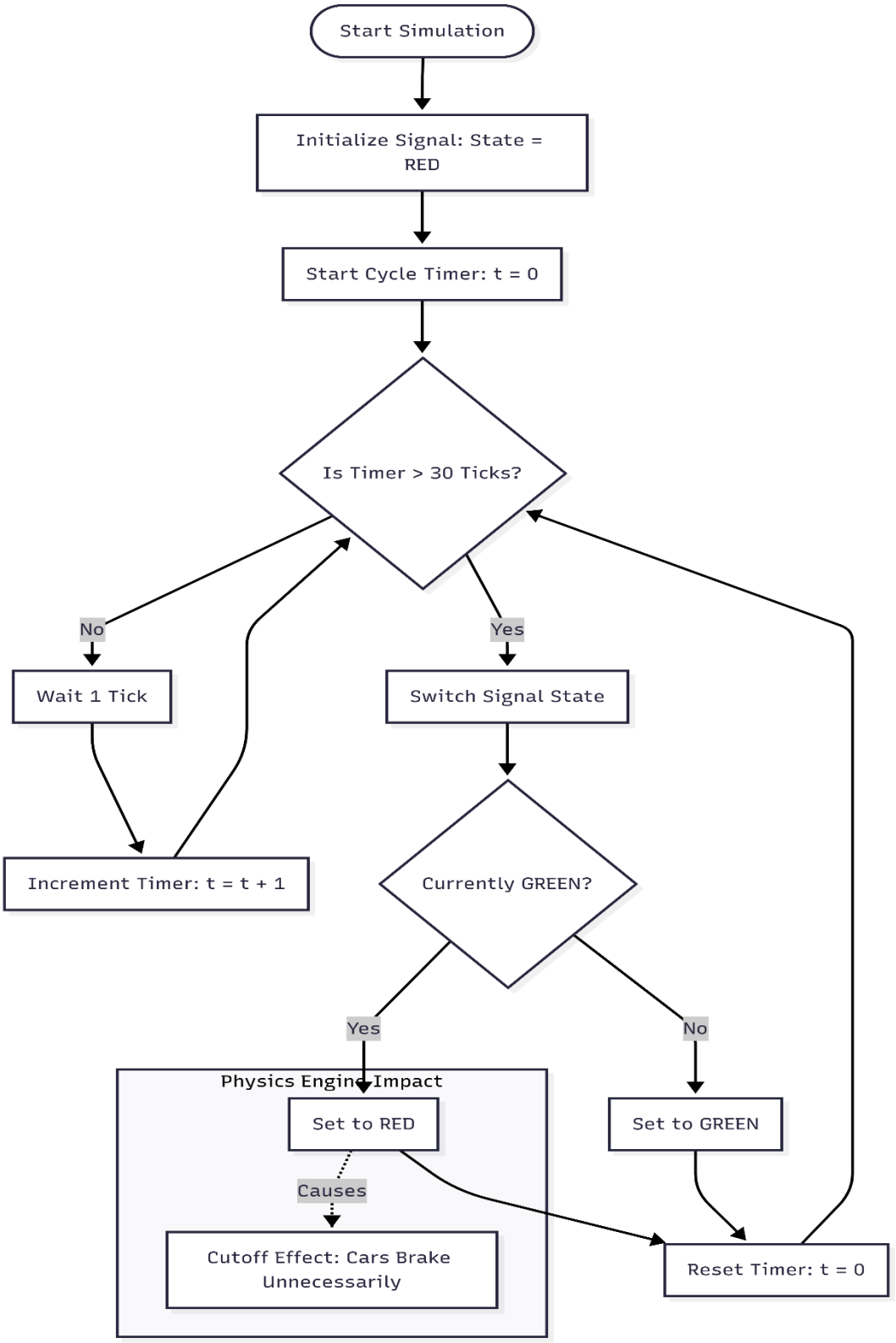
5.1.3 Topology repair

```
; Road Network Repair (The "Pothole Filler")

  ; Clean edges
  ask patches [
    if (pxcor <= min-pxcor + 2) or (pxcor >= max-pxcor - 2) or
       (pycor <= min-pycor + 2) or (pycor >= max-pycor - 2) [
      set pcolor white
    ]
  ]

  ; Bake drawing to patches
  export-view "temp_map.png"
  import-pcolors "temp_map.png"
  clear-drawing

  ; Define drivable surfaces
  ask patches with [pcolor >= 125 and pcolor <= 135] [ set is-road? true ]

  ; Fix rendering artifacts (The "Black Patch" bug)
  ask patches with [pcolor = 0] [ set pcolor 5 ]
```

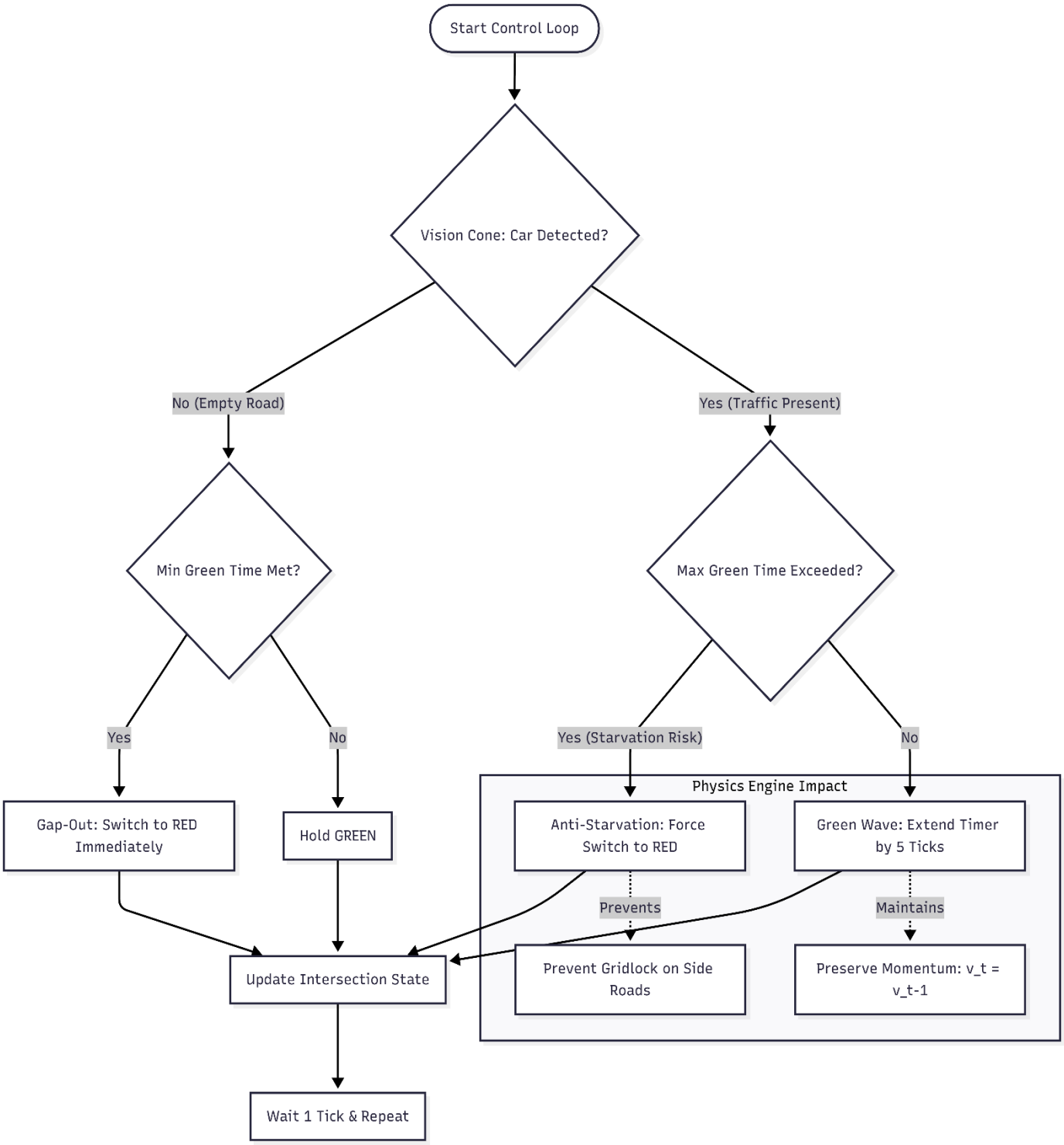
```
; Algorithm: Intelligent Road Healer
; Looks for disconnected road pixels and bridges them
repeat 5 [
  ask patches with [pcolor = 5] [
    if count neighbors with [is-road?] >= 4 [
      set pcolor 130
      set is-road? true
    ]
  ]
]

; Algorithm: Gap Closer
ask patches with [not is-road? and pcolor != 5] [
  if count neighbors with [is-road?] >= 5 [
    set is-road? true
    set pcolor 130
  ]
]
```

5.2 Algorithm logic (Flow charts)

5.2.1 Static mode logic



5.2.2 Dynamic mode logic



5.3 Gis verification (Digital Twin)

Comparison of real-world topology vs. the generated Digital Twin."

A screenshot of your NetLogo "World View" showing the same area.

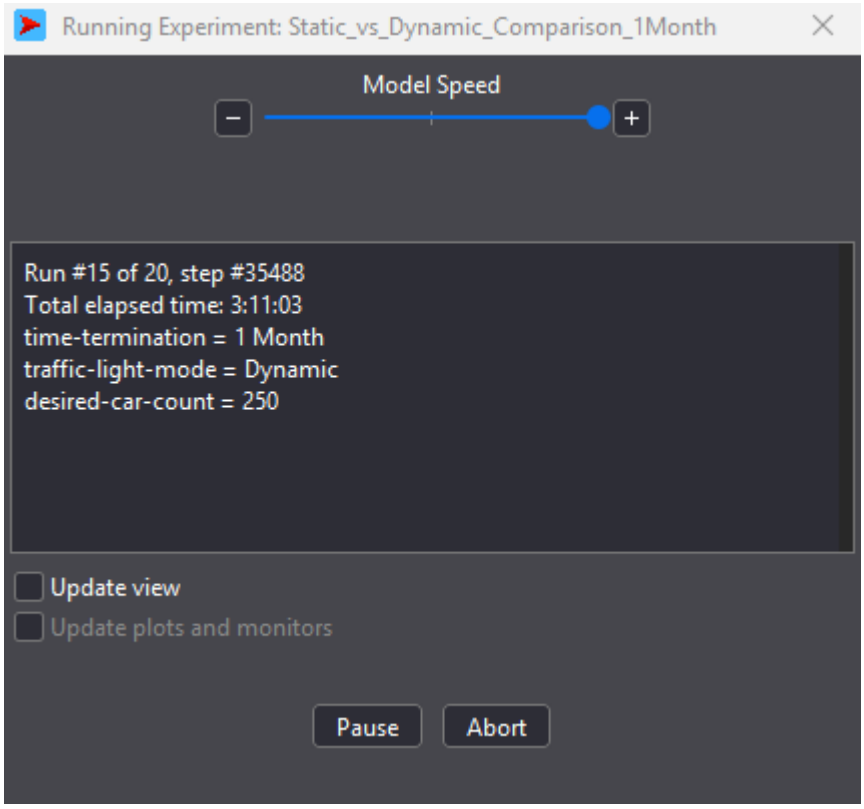
A screenshot of the real area in Leicester from Google Maps

5.4 Raw data samples (Evidence)

Mode	Cars	Duration	Total_CO2	Avg_Speed	Trips_Done
Static	250	1 Day	9363.9	28.74	122
Static	250	1 Day	9774.54	28.31	128
Static	250	1 Day	9432.9	28.69	123
Static	250	1 Day	9639.48	27.94	117
Static	250	1 Day	9275.13	28.62	121
Static	250	1 Day	9713.45	28.48	127
Static	250	1 Day	9411.15	28.56	120
Static	250	1 Day	9394.81	28.44	122
Static	250	1 Day	9501.23	28.45	131
Static	250	1 Day	9994.8	28.46	125
Dynamic	250	1 Day	9174.62	29.73	116
Dynamic	250	1 Day	9019.49	28.85	124
Dynamic	250	1 Day	9065.01	29.29	120
Dynamic	250	1 Day	9483.51	28.98	119
Dynamic	250	1 Day	9134.83	28.82	123
Dynamic	250	1 Day	9911.83	28.77	127
Dynamic	250	1 Day	9137.64	29.28	119
Dynamic	250	1 Day	9468.1	29.02	124
Dynamic	250	1 Day	9218.13	28.99	127
Dynamic	250	1 Day	9380.97	28.36	114
Static	250	1 Week	91787.48	28.81	1463
Static	250	1 Week	89901.47	28.49	1458
Static	250	1 Week	90478.03	28.34	1494
Static	250	1 Week	90066.82	28.54	1501
Static	250	1 Week	92721.32	29.5	1496
Static	250	1 Week	90778.42	29.26	1515
Static	250	1 Week	90345.1	29.29	1407
Static	250	1 Week	92434.78	28.34	1439
Static	250	1 Week	91560.5	28.24	1410
Static	250	1 Week	93018.77	28.73	1399
Dynamic	250	1 Week	87372.39	28.94	1475
Dynamic	250	1 Week	89479.69	28.84	1328
Dynamic	250	1 Week	89710.42	29.7	1528
Dynamic	250	1 Week	88933.68	28.8	1556
Dynamic	250	1 Week	85891.1	28.24	1471
Dynamic	250	1 Week	87995.31	29.45	1483
Dynamic	250	1 Week	85443.48	28.99	1520
Dynamic	250	1 Week	89816.05	29.11	1463
Dynamic	250	1 Week	84624.06	29.44	1493
Dynamic	250	1 Week	90150.1	29.36	1530
Static	250	1 Month	394617.73	28.32	6489
Static	250	1 Month	405157.54	28.98	6323

Static	250	1 Month	412052.09	28.84	6386
Static	250	1 Month	412191.47	30.21	6319
Static	250	1 Month	412524.37	27.36	6256
Static	250	1 Month	414076.6	29.18	6318
Static	250	1 Month	418196.04	29.65	6248
Static	250	1 Month	421418.86	29.58	6216
Static	250	1 Month	421661.57	28.72	6336
Static	250	1 Month	421805.24	28.08	6069
Dynamic	250	1 Month	384193.84	29.38	6348
Dynamic	250	1 Month	387231.27	29.74	6325
Dynamic	250	1 Month	389099.76	29.5	6342
Dynamic	250	1 Month	390207.97	28.91	6358
Dynamic	250	1 Month	391545.04	29.34	6392
Dynamic	250	1 Month	391616.81	29.97	6343
Dynamic	250	1 Month	392457.78	29.32	6502
Dynamic	250	1 Month	397147.77	28.99	6268
Dynamic	250	1 Month	398417.92	29.69	6179
Dynamic	250	1 Month	400843.82	29.35	6195

5.5 Multithreading behaviour configuration



Parameters Table:

Variables Varied: ['Static', 'Dynamic']

Repetitions: 10

Time Limit: 43,200 ticks (1 Month)

Parallel Processors: 14

This ensured running 20 experiments in the duration of 1 experiment.

5.6 Video recording of a single simulation

Parameters:

Car count: 250

Duration: 1 Day

Link to the video