TaxiGreen

AI-Driven Fleet Optimization

Universidade do Minho | Grupo 06

João Delgado | Nelson Mendes | Simão Mendes | Tomás Machado

Presentation Agenda

01. Problem Description: System representation (S,A,T,C) and fleet management.

02. City Representation: Braga's graph, zonal classification, and critical infrastructure.

03. Search Strategies: Comparison between Uninformed and Informed search algorithms.

04. Dynamic Simulation: Real-time traffic, weather modeling, and random request generation.

05. Performance Benchmark: Quantitative analysis of time, memory, and path optimality.

06. Architecture & Results: System design, multithreaded server, and final conclusions.

The Challenge: Sustainable Mobility

Goal: Optimize a mixed fleet (combustion & electric) in a real urban environment.

- Manage distinct constraints: Range anxiety & charging times.
- Operate within the complex topology of Braga, Portugal.
- Balance conflicting objectives: Minimizing time vs. Maximizing sustainability.

Problem Formalization (S, S_0, A, T, C)

State (S)

Vehicle Location , Battery Level , Operational Status , Passenger Capacity , Engine Type , Pending Requests , Priority , Deadline , Environmental Preference , and Dynamic Traffic Conditions.

Actions (A)

Move(v, dest), Pickup(v, p), Dropoff(v, p), Refuel(v, s).

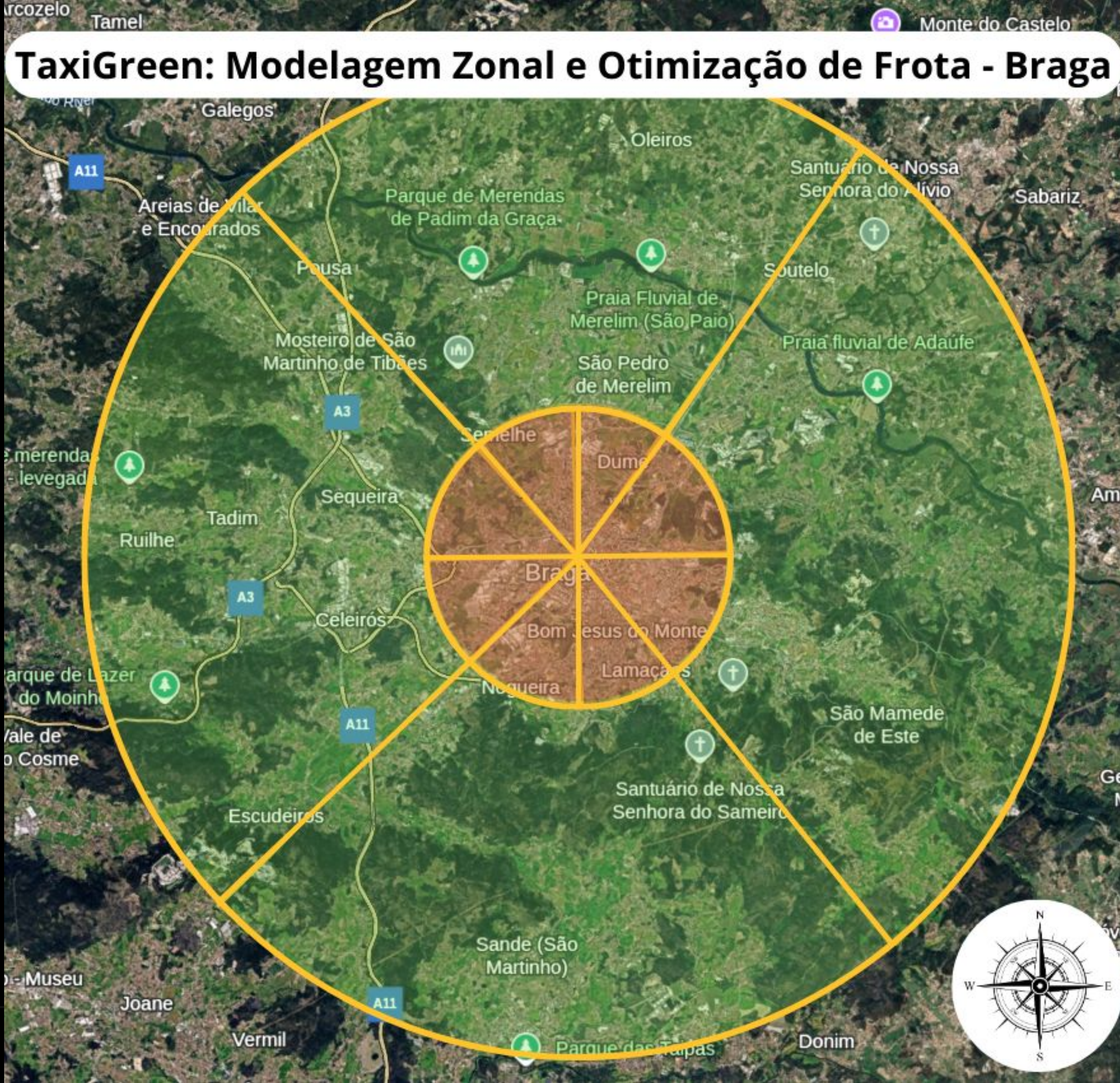
Goal (T)

Pending Requests Empty ($P=\emptyset$) , Successful Passenger Transport , Deadlines Met , and Guaranteed Vehicle Autonomy.

Modeling Braga

We modeled the entire municipality using **OSMnx** and OpenStreetMap data.

- **Graph:** 8,624 Nodes & 19,753 Edges.
- **Zonal Logic:**
  - Center: High density, 8 cardinal sectors (Traffic heavy).
  - Periphery: Rural, 4 sectors (Weather sensitive).
- **Ecological Score:** dynamic (η) for each road segment.



The Eco-Score

Calculating Sustainability

Eco-Score Definition Dynamic Rating $\eta \in [0,1]$, Environmental Weight (β=0.2), and Route Sustainability

- 🍃 **High Score (0.9 - 1.0):** Residential streets, cycle lanes, low-speed zones.
- ⚠️ **Medium Score (0.5 - 0.8):** Main avenues, secondary distributors.
- 🏭 **Low Score (0.0 - 0.4):** Highways, industrial zones, high-speed arterials.

🌲

Effect on Routing

An edge with $\eta = 1$ incurs **zero penalty**.

An edge with $\eta = 0$ increases cost by **20%** (when $\beta = 0.2$).

Cost Function

Dynamic Cost Function

We calculate the cost of traversing an edge (u, v) by weighting time with environmental factors.

$$c(u, v) = [t_{base}(u, v) \times M_{climate}(u) \times M_{traffic}(u)] \times [1 + \beta \times (1 - \eta(u, v))]$$

Simplified Dynamic Cost Function

$$Cost_{total} = Cost_{base} \times M_{climate} \times M_{traffic} \times Score_{Eco}$$

Where:

- **t_{base} (u, v):** represents the **minimum travel time** (in seconds), calculated by the ratio of the edge length to its maximum speed converted to m/s;
- **$M_{climate}$** e **$M_{traffic}$**: are **dynamic coefficients (≥ 1.0) determined by the geographical area** of the node of origin;
- **β = 0.2:** is the **weight given to the ecological component** of the system;
- **η(u,v) ∈ [0,1]:** is the **Eco-Score** of the edge, where higher values indicate more environmentally friendly routes.

Heuristic

Admissible Heuristic

To ensure A* optimality, we use the Haversine distance divided by the global maximum speed.

$$h(u, v) = \frac{D_{\text{Haversine}(u,v)}}{\frac{v_{\text{global\_max}}}{3.6}}$$

Where:

- $D_{\text{Haversine}}(u,v)$: Calculates the great-circle distance (straight line) between the current node and the destination.
- $v_{\text{global\_max}}$: Represents the highest speed limit recorded in the entire graph (e.g., 120 km/h on highways).
- 3.6 Conversion Factor: Used to convert speed from km/h to m/s to ensure metric consistency.

Search Strategies

Uninformed Search

DFS: Explores deeply before backtracking. Gets lost in cycles or long paths. Result: ≈ +2000% Cost.

BFS: Guarantees fewest hops, but ignores traffic/eco costs. Result: ≈ +31.8% higher cost.

UCS: Guarantees optimal cost, but explores radially (slow ≈ 63.30ms).

Informed Search

Greedy: Fastest (≈ 3.05ms), but suboptimal solutions (≈ +27.5% cost).

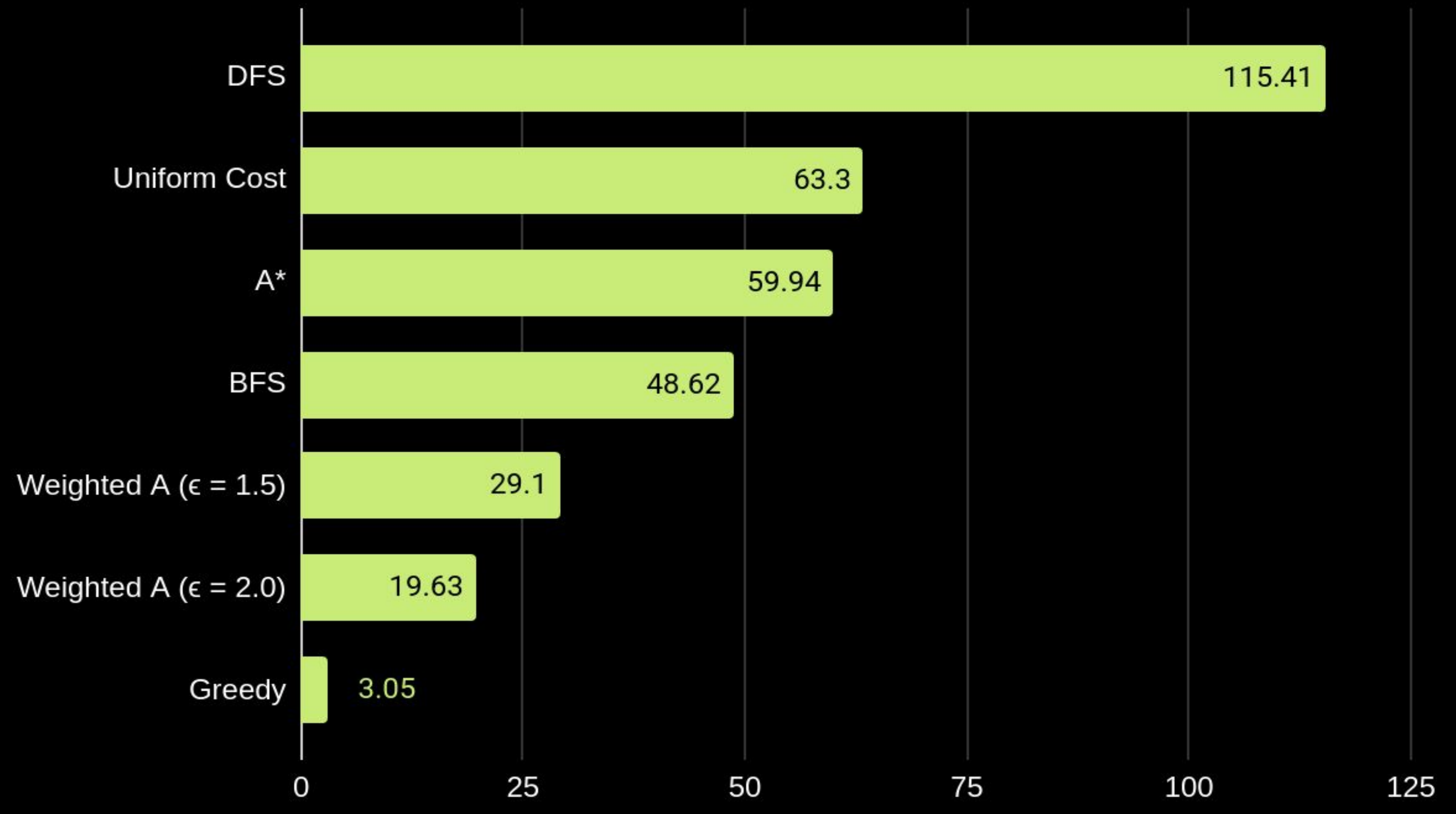
A* (A-Star): The gold standard. Combines UCS optimality with Greedy's direction.

Weighted A*: Prioritizes moving towards goal over minimizing path cost. The Winner (≈ 19ms with $\varepsilon = 2.0$).

Benchmarking Methodology

- **Isolated Environment:** Tests ran on a dedicated script, bypassing the UI/Network latency.
- **100 Iterations:** We generated 100 random (Origin, Destination) pairs.
- **Consistency:** Exactly 100 iterations per route to ensure results comparability.
- **Tools:** Python time.perf_counter() and tracemalloc for memory profiling.

Benchmarking: Execution Time (milliseconds)



Greedy is the fastest but not the best due to suboptimal results; however, A is the top performer, as shown in the next slide.*

The Trade-off: Speed vs. Optimality

Algoritmo	Sucesso	Tempo (ms)	Memória (KB)	Custo	Passos	Eficiência
A*	100%	**59.54**	504.5	**559.2**	65.9	**Ótimo**
Uniform Cost	100%	63.30	670.4	559.2	65.9	**Ótimo**
Weighted A* ($\varepsilon = 1.5$)	100%	29.10	738.3	559.2	65.9	**Ótimo**
Weighted A* ($\varepsilon = 2.0$)	100%	**19.63**	752.7	559.2	65.9	**Ótimo**
Greedy	100%	**3.05**	**91.8**	712.7	71.6	+27.5%
BFS	100%	48.62	418.2	737.1	**44.1**	+31.8%
DFS	100%	115.41	23,776.8	12,056.7	1314.0	+2056%

Optimization: Heuristic Caching

The computational cost of the Haversine formula initially slowed down A*. Caching solved this.

84ms
PRE-OPTIMIZATION

29.1%
PERFORMANCE BOOST

≃59ms
POST-OPTIMIZATION

The Trade-off Analysis

The "Golden Mean"

We selected **Weighted A* (ε = 2.0)** as our primary engine.

- **Speed:** 19.63ms (Extremely fast)
- **Cost:** 559.2 (Identical to Optimal)
- **Why?** In the topology of Braga, the slight "greediness" (ε = 2.0) directs the search efficiently without getting trapped in suboptimal local minima.Weighted

3x

FASTER THAN A*

0%

COST PENALTY

System Architecture

Server

Multithreaded TCP Server (Port 8888). Maintains simulation loop, by sending a random request every 3 seconds.

Client

One-threaded client using socket and pickle for serialization. Decoupled from logic.

Shared State

Thread-safe locks protect the Fleet and Request Queue during concurrent access.

Rich Terminal UI

Developed using the **Python Rich** library, the interface offers a dashboard-like experience in the terminal.

- ✓Live Fleet Status Tables
- ✓Algorithm Switcher
- ✓Real-time System Logs
- ✓"Matrix" Aesthetic

Dynamic Simulation

The system is not static. It reacts to a chaotic environment to test fleet resilience.

- **Variable Traffic:** Zonal multipliers simulate rush hours ($M_{traffic}$ > 2.0$).
- **Weather Events:** Rain reduces average speed.
- **Infrastructure Failure:** Charging stations or streets in general, can go down, forcing re-routing.

Conclusion

- **Modeling Success:** Graph representation with zonal logic effectively captures Braga's complexity.
- **Algorithm Winner:** Weighted A* (ε=2.0) offers the best balance of speed and optimality for real-time dispatch.
- **Sustainability:** The dynamic cost function successfully prioritizes eco-friendly routes without compromising service levels.
- **Technical Robustness:** Threaded architecture ensures smooth operation under load.

Q&A

Thank you for your attention.

```
root@taxigreen:~$ shutdown -h now_
```

TaxiGreen

AI-Driven Fleet Optimization

Universidade do Minho | Grupo 06

João Delgado | Nelson Mendes | Simão Mendes | Tomás Machado