Moving Mountains of Parcels with the Mathematics of Earth Moving

Optimal Transport Theory Applied to Indonesian Logistics Networks

By Hilmi

Research Areas: Optimal Transport · Convex Optimisation · Logistics

Key Methods: Kantorovich LP · Sinkhorn Algorithm · Dynamic Temporal Coupling

Applications: Indonesian Supply Chains · Seasonal Demand Planning · Fairness Analysis

Project Statement

My project sits at the intersection of **mathematical optimisation**, **machine learning**, and **real-world decision systems** in the Indonesian context. I am driven by a single question: *How can we make algorithms that are not only theoretically sound but also practically deployable in emerging-economy logistics?*

Optimal transport is a beautiful example of this tension. The classical Kantorovich formulation gives us the mathematically "correct" way to route goods from warehouses to customers but it produces a single static plan that ignores the operational reality of day-to-day stability. Meanwhile, the popular Sinkhorn algorithm trades exactness for speed via entropic regularisation, but does not address the temporal dimension at all. And neither framework asks the question that matters most during Indonesia's Ramadan surge: *when there are not enough goods for everyone, who gets served?*

This project attacks these problems head-on by combining **five optimal transport formulations** (classical, entropic, unbalanced, partial, and graph-augmented) with a **novel multi-period temporal coupling extension**. The temporal formulation adds a Frobenius norm penalty between consecutive transport plans, encouraging route stability while maintaining near-optimal cost. Solved via block-coordinate proximal Sinkhorn iterations with gradient clipping, it achieves 3–8% stability improvement at less than 0.1% cost increase, a trade-off any logistics manager would accept.

Why Indonesia? The archipelago presents logistics challenges that stress-test any framework: 17,000 islands, dramatic demand seasonality (Ramadan and Harbolnas can shift supply-to-demand ratios from 1.2 to 0.1), and stark urban-rural service disparities. These are not theoretical edge cases, they are the daily reality for Indonesian logistics operators. By testing on three networks (Jabodetabek metropolitan, Java intercity, and archipelago-wide), this project demonstrates that optimal transport theory is not just mathematically elegant but operationally useful.

Broader research trajectory. This project reflects my systematic approach: begin with a real problem, connect it to deep mathematical structure, implement state-of-the-art algorithms, and validate rigorously. My other work follows this pattern across graph neural networks for fraud detection, causal inference for Indonesian social protection policy, and NLP for Indonesian-specific applications.

What I learned. Perhaps the most valuable outcome was discovering through systematic debugging that the multi-period solver required three sequential fixes before producing valid results. Finding these bugs (a missing gradient factor, biased cost clipping, and scale-dependent instability) required deeper engagement with proximal algorithm theory than simply "making the code run." This experience reinforced my belief that **reproducible research requires more than reproducible code** it requires someone who understands the theory well enough to know when the numbers are wrong.

Contents

1 The Business Problem: Why Logistics Needs Better Mathematics

1.1 The Logistics Manager's Daily Challenge

Imagine you manage Jabodetabek Express, a delivery company operating in greater Jakarta. You have 12 warehouses, each holding a known quantity of product, and 200 retail stores and fulfilment points, each with specific daily demand. Your job: decide how much product to ship from each warehouse to each store, every single day, at minimum cost.

This is not a hypothetical problem. Indonesian logistics companies make these decisions daily, coordinating thousands of routes across a network that spans dense urban centres, intercity highways, and cross-island sea lanes. The mathematics of optimal transport gives you the cheapest plan for any single day. But a logistics manager needs more than just today's cheapest plan:

- **Route stability:** Drivers perform better on familiar routes. Loading dock schedules depend on predictable outbound flows. If the "optimal" plan changes 40% of routes every day, the theoretical savings evaporate in operational chaos. *Can mathematics give us plans that are cheap AND stable over time?*

- **Demand surges:** During Ramadan, food demand can surge 2.5×. During Harbolnas (Indonesia's national online shopping day), electronics demand can spike 4.5×. When your 12 warehouses cannot possibly serve all 200 stores, you must decide who gets served and who does not. *Can mathematics make this allocation fair?*

- **Geographic equity:** A cost-minimising algorithm naturally favours dense urban areas where delivery is cheap, potentially leaving rural communities with 6% fulfilment while urban customers get 92%. *Can mathematics quantify and address this inequity?*

- **Road network reality:** Two stores may be equidistant from a warehouse as the crow flies, but one is connected by a highway and the other by a mountain road. *Can mathematics incorporate actual road networks?*

The Core Questions of This Project

This project addresses four business questions using optimal transport theory:

1. **Cost:** What is the cheapest way to route goods? (Classical OT — Section 2.2)
2. **Stability:** How can we keep routes stable over time without sacrificing cost? (Multi-Period OT — Section 2.7)
3. **Fairness:** When demand exceeds supply, who gets served? (Partial and Unbalanced OT — Section 2.5)
4. **Realism:** How do road networks change the optimal routes? (Graph-Augmented OT — Section 2.6)

Each question maps to a mathematical formulation, an implementation, and one or more experiments.

1.2 A Concrete Example: Why "Just Minimise Cost" Is Not Enough

Consider a simplified example. You have 3 warehouses (W1 in Jakarta, W2 in Bogor, W3 in Tangerang) and 6 stores. On Monday, the cheapest plan routes W1 to Stores 1–3, W2 to Stores 4–5, and W3 to Store 6. On Tuesday, demand shifts slightly: Store 4 needs more, Store 3 needs less. The new cheapest plan now routes W1 to Stores 1, 2, and 5, W2 to Stores 3–4, and W3 to Store 6. Three routes have changed.

For a spreadsheet, this is a trivial update. For a warehouse manager, it means re-assigning drivers, changing loading sequences, updating customer expectations, and adjusting last-mile vehicle allocations. Multiply this by 30 days and 200 stores, and the cost of operational disruption can dwarf the transport cost savings from daily re-optimisation.

The multi-period formulation (Section 2.7) solves this by adding a penalty for changing routes between days. The result: plans that are 3–8% more stable while costing less than 0.1% more. This is the kind of trade-off that makes mathematics useful to logistics.

1.3 Quantifying the Impact

Table 1 summarises the practical impact across the four business questions.

1.4 Beyond Logistics: Where These Methods Apply

The optimal transport framework extends far beyond physical parcel delivery:

1. **Supply Chain Risk Management:** Unbalanced OT quantifies how gracefully a supply chain degrades under disruption (pandemic, natural disaster, port closure).
2. **Healthcare Resource Allocation:** During a pandemic, partial OT determines

Table 1: Business impact summary across all seven experiments.

Business Question	**Method**	**Key Finding**
Minimise transport cost	Classical OT (LP)	Uses exactly $n + m - 1$ routes (211 out of 2,400 for Jabodetabek)
Improve route stability	Multi-Period OT ($\lambda = 0.5$)	3–8% stability gain at <0.1% cost increase, validated across 5 seeds
Handle demand surges	Unbalanced OT	Graceful degradation under Ramadan scarcity (S/D ≈ 0.4)
Fair allocation	Partial OT under scarcity	Reveals 7–14× urban/rural disparity on Java intercity
Road network realism	Graph-Augmented Cost	12–77% cost impact depending on network topology

how to distribute limited vaccines or ventilators across hospitals with different needs, the fairness analysis of Experiment 7 applies directly.

3. **Humanitarian Logistics:** After a natural disaster, warehouses become relief depots and customers become affected communities. The Ramadan scarcity scenario (S/D ≈ 0.4) models exactly this situation.

4. **Retail Inventory Positioning:** Seasonal demand patterns (Ramadan, Harbolnas) require repositioning inventory across a network. The multi-period formulation provides smooth transitions between seasonal configurations.

5. **Energy Grid Balancing:** Generation plants are "warehouses" and consumption centres are "customers." OT finds the optimal power flow, and temporal coupling ensures grid stability during demand ramps.

2 Mathematical Foundations

This section builds the theory from the ground up. We start with Monge's 1781 problem, progress through Kantorovich's relaxation, introduce five modern formulations, and arrive at the novel multi-period extension. Every concept is connected to its logistics interpretation.

2.1 The Monge Problem (1781): Moving Soil

The oldest formulation of optimal transport was posed by Gaspard Monge in 1781: given a pile of sand (source distribution μ) and a hole to fill (target distribution ν), find the

cheapest way to move the sand.

Definition 2.1 (Monge Problem)**.** Given source measure μ on $\mathcal{X}$, target measure ν on $\mathcal{Y}$, and cost function $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$, find a transport map $T : \mathcal{X} \to \mathcal{Y}$ such that $T_{\#}\mu = \nu$ (mass is conserved) minimising:

$$\inf_{T:T_{\#}\mu=\nu} \int_{\mathcal{X}} c(x, T(x))\, d\mu(x)$$

Logistics interpretation: Each warehouse ships *all* its goods to exactly one store. You cannot split a warehouse's inventory. This is sometimes realistic (a small depot with one truck), but usually too restrictive for modern distribution networks.

The Monge formulation has a fundamental limitation: it requires each source point to map to exactly one destination. This makes the problem non-convex and sometimes infeasible (e.g., if there are more stores than warehouses).

2.2 The Kantorovich Relaxation (1942): Allowing Splitting

Leonid Kantorovich's insight was to allow mass splitting: a warehouse can ship to multiple stores simultaneously. Instead of a map T, we seek a *coupling matrix* (or transport plan) $\mathbf{T}$ where T_{ij} represents the amount shipped from warehouse i to store j (Kantorovich, 1942).

Definition 2.2 (Kantorovich Problem)**.** Given supply $\mathbf{a} \in \mathbb{R}^n_+$, demand $\mathbf{b} \in \mathbb{R}^m_+$ with $\sum_i a_i = \sum_j b_j$, and cost matrix $\mathbf{C} \in \mathbb{R}^{n\times m}_+$, the transportation polytope is:

$$\Pi(\mathbf{a}, \mathbf{b}) = \left\{ \mathbf{T} \in \mathbb{R}^{n\times m}_+ \;\middle|\; \mathbf{T}\mathbf{1}_m = \mathbf{a},\ \mathbf{T}^\top \mathbf{1}_n = \mathbf{b} \right\}$$

The Kantorovich problem is:

$$\mathbf{T}^* = \arg\min_{\mathbf{T}\in\Pi(\mathbf{a},\mathbf{b})} \langle \mathbf{C}, \mathbf{T} \rangle_F = \arg\min_{\mathbf{T}\in\Pi(\mathbf{a},\mathbf{b})} \sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij}T_{ij} \tag{1}$$

Logistics interpretation: The constraints $\mathbf{T}\mathbf{1} = \mathbf{a}$ say "every warehouse ships out exactly its capacity." The constraints $\mathbf{T}^\top\mathbf{1} = \mathbf{b}$ say "every store receives exactly its demand." The objective minimises total shipping cost.

Theorem 2.1 (LP Sparsity / Birkhoff-von Neumann Bound)**.** Every vertex of the transportation polytope $\Pi(\mathbf{a}, \mathbf{b})$ has at most $n+m-1$ non-zero entries. Since the LP optimum is attained at a vertex, the optimal transport plan uses at most $n + m - 1$ routes out of $n \times m$ possible.

Logistics interpretation: With 12 warehouses and 200 customers, the optimal plan uses at most 211 routes out of 2,400 possible, **91.2% of routes are unused**. This sparsity is operationally attractive: each warehouse ships to only a handful of stores, making the plan implementable. Experiment 1 confirms this bound exactly.

This is a linear programme and is solved exactly using the network simplex algorithm (Flamary et al., 2021), implemented in the POT library.

2.3 Entropic Regularisation: The Sinkhorn Algorithm

While the classical LP is exact, it can be slow for large problems and is not differentiable (important for machine learning applications). Cuturi (2013) introduced entropic regularisation:

$$\mathbf{T}_\varepsilon^* = \arg\min_{\mathbf{T}\in\Pi(\mathbf{a},\mathbf{b})} \langle \mathbf{C}, \mathbf{T}\rangle_F - \varepsilon H(\mathbf{T}) \tag{2}$$

where $H(\mathbf{T}) = -\sum_{ij} T_{ij}(\log T_{ij} - 1)$ is the negative entropy (the KL divergence from the Lebesgue measure), following the precise formulation in Peyré and Cuturi (2019), Section 4.2.

The solution has the form $T_{ij}^* = u_i K_{ij} v_j$ where $K_{ij} = \exp(-C_{ij}/\varepsilon)$ is the Gibbs kernel. The vectors $\mathbf{u}$ and $\mathbf{v}$ are found by the Sinkhorn-Knopp algorithm: alternately normalising rows and columns of K.

The Regularisation–Sparsity Trade-off for Logistics Managers

Think of ε as a "route diversification dial":

- **Small ε (e.g., 0.001):** Nearly optimal cost, almost as sparse as the LP. A few heavily loaded routes. Cost-efficient but fragile if one route fails, there is no backup.
- **Large ε (e.g., 1.0):** Higher cost, many lightly loaded routes. Every warehouse ships a little bit to every store. Resilient but expensive and operationally impractical.
- **Sweet spot ($\varepsilon \approx 0.01$–0.1):** Moderate cost increase (1–5%) with enough route diversification for operational resilience.

Experiment 1 sweeps $\varepsilon \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0\}$ to map this trade-off precisely across all three Indonesian networks.

2.4 Unbalanced Optimal Transport: When Supply Does Not Meet Demand

Classical OT requires $\sum_i a_i = \sum_j b_j$. During Ramadan, total demand may exceed total supply by 2.5×. The unbalanced OT formulation of Chizat et al. (2018) relaxes the marginal constraints using KL divergence penalties:

$$\mathbf{T}^* = \arg\min_{\mathbf{T}\geq 0} \langle \mathbf{C}, \mathbf{T}\rangle_F + \varepsilon\,\mathrm{KL}(\mathbf{T}\|K) + \tau\,\mathrm{KL}(\mathbf{T}\mathbf{1}\|\mathbf{a}) + \tau\,\mathrm{KL}(\mathbf{T}^\top\mathbf{1}\|\mathbf{b}) \tag{3}$$

where ε is the entropic regularisation and τ (denoted `reg_m` in the code) controls marginal enforcement strictness.

Logistics interpretation: When τ is large, the solver insists on satisfying as much demand as possible (even at high cost). When τ is small, the solver accepts partial

service to keep costs down. The parameter τ is the logistics manager's knob for "how hard do we try to serve everyone?"

Business Decision: The τ Parameter

A logistics VP faces this decision every Ramadan: "We have capacity for 40% of demand. Do we (a) serve the 40% cheapest-to-reach customers fully, or (b) give everyone 40% of what they need?" Unbalanced OT with varying τ gives you a smooth continuum between these extremes, letting you find the right balance between cost efficiency and service coverage.

2.5 Partial Optimal Transport: When Capacity Is Limited

Partial OT (Chapel et al., 2020) provides an alternative model for capacity constraints: transport exactly m units of mass, where $m < \min(\sum_i a_i, \sum_j b_j)$.

$$\mathbf{T}^* = \arg\min_{\mathbf{T} \geq 0} \langle \mathbf{C}, \mathbf{T} \rangle_F \quad \text{s.t.} \quad \mathbf{T}\mathbf{1} \leq \mathbf{a}, \quad \mathbf{T}^\top \mathbf{1} \leq \mathbf{b}, \quad \mathbf{1}^\top \mathbf{T} \mathbf{1} = m \tag{4}$$

The inequality constraints allow partial utilisation of both supply and demand. The solver chooses the cheapest subset of supply-demand pairs to transport exactly m units.

Logistics interpretation: "We only have trucks for 70% of deliveries today. Which 70% should we serve?" The answer: partial OT picks the cheapest-to-serve 70%, which means urban customers near warehouses get prioritised and rural customers get left out. This is the mechanism behind the urban-rural equity gaps documented in Experiment 7.

2.6 Graph-Augmented Cost Matrices

Standard OT uses Euclidean (straight-line) distance. In reality, goods move along road networks. Following Vayer et al. (2019), this project computes shortest-path distances using Dijkstra's algorithm and constructs a hybrid cost:

$$C_{\text{hybrid}} = (1 - \alpha)\, C_{\text{Euclidean}} + \alpha\, C_{\text{graph}} \tag{5}$$

Logistics interpretation: At $\alpha = 0$, you pretend roads go in straight lines. At $\alpha = 1$, you use actual road distances. The reality for Indonesian logistics is that two points on different islands may be 50 km apart as the crow flies but 500 km apart by the nearest sea-road route. Experiment 5 shows this matters enormously for the Archipelago dataset (77% cost difference) but less for dense urban networks (12% for Jabodetabek).

2.7 Multi-Period Dynamic Optimal Transport (Novel Contribution)

All formulations above solve a single-period problem: given today's supply and demand, find today's plan. This project introduces a temporally-coupled formulation that jointly optimises plans across P consecutive periods:

$$\min_{T_1,\dots,T_P} \sum_{t=1}^{P} \Big[\langle C_t, T_t\rangle_F + \varepsilon\, H(T_t)\Big] + \lambda \sum_{t=2}^{P} \|T_t - T_{t-1}\|_F^2 \tag{6}$$

The first term minimises cost for each day. The second term, the **temporal coupling penalty**, penalises large changes between consecutive days' plans. The parameter $\lambda \geq 0$ controls how much the manager values stability over cost.

Logistics interpretation: "I want to minimise shipping cost, but I also want tomorrow's routes to look similar to today's routes. How much am I willing to pay for that stability?" At $\lambda = 0.5$, the answer is: "almost nothing" (less than 0.1% cost increase for 3–8% stability improvement).

2.7.1 Solving via Block-Coordinate Proximal Sinkhorn

Direct optimisation of Equation 6 is intractable because the temporal coupling links all P sub-problems. We use a block-coordinate proximal approach inspired by Parikh and Boyd (2014):

Algorithm 1 Block-Coordinate Proximal Sinkhorn for Multi-Period OT

1: **Phase 1 (Initialise):** For each period $t = 1, \dots, P$, solve independent Sinkhorn:
2: $T_t^{(0)} = \text{Sinkhorn}(\mathbf{a}_t, \mathbf{b}_t, C_t, \varepsilon)$
3: **for** outer iteration $k = 1, 2, \dots, K_{\max}$ **do**
4: **for** period $t = 1, \dots, P$ **do**
5: Compute bilateral anchor: $\text{anchor}_t = \frac{1}{|\mathcal{N}(t)|} \sum_{s \in \mathcal{N}(t)} T_s^{(k)}$
6: Compute proximal gradient: $G_t = 2\lambda(T_t^{(k)} - \text{anchor}_t)$
7: **Clip gradient:** $G_t \leftarrow \text{clip}(G_t, -0.5 \cdot \Delta C, +0.5 \cdot \Delta C)$ where $\Delta C = \max(C) - \min(C)$
8: Augment cost: $C_{\text{aug}} = C_t + G_t$
9: **Shift if needed:** $C_{\text{aug}} \leftarrow C_{\text{aug}} - \min(C_{\text{aug}})$ if any entry < 0
10: Solve: $T_t^{(k+1)} = \text{Sinkhorn}(\mathbf{a}_t, \mathbf{b}_t, C_{\text{aug}}, \varepsilon)$
11: **end for**
12: Check convergence: stop if relative cost change $< 10^{-6}$
13: **end for**

Why Gradient Clipping Matters — A Debugging Story

On the Java intercity network (only 8 warehouses), the proximal gradient at $\lambda = 1.0$ can be $10\times$ larger than the base cost entries. Without clipping, the Sinkhorn solver sees a cost matrix dominated by the temporal term rather than actual transport costs, causing it to produce nonsensical plans. The gradient clipping in Step 7 caps the temporal influence at 50% of the cost range, preserving gradient direction while preventing numerical instability. See Section 7 for the full debugging journey.

Proposition 2.1 (Convergence)**.** The block-coordinate proximal scheme in Algorithm 1 is an instance of the Gauss-Seidel proximal point algorithm. Under the condition that λ is bounded relative to ε and the gradient clipping threshold is positive, the iterates

converge to a stationary point of the joint objective (Parikh and Boyd, 2014; Bauschke and Combettes, 2017).

The Uniform Shift Trick (Step 9)

Sinkhorn requires non-negative cost matrices ($K_{ij} = \exp(-C_{ij}/\varepsilon)$ must be well-defined). When the proximal gradient creates negative entries in C_{aug}, we shift the entire matrix up by $|\min(C_{\text{aug}})|$. This is mathematically valid because a uniform additive shift to the cost matrix does not change the Sinkhorn optimal plan, it affects only the dual variables, not the primal solution. This is a standard result in LP theory (adding a constant to the objective does not change the optimizer).

3 From Theory to Implementation: How the Mathematics Becomes Code

This section bridges the mathematical foundations of Section 2 and the concrete code in Sections 4–6. For every theoretical concept, we show *exactly* how it translates into an algorithmic design decision, which file implements it, and which experiment validates it.

3.1 The Kantorovich Problem → The Classical OT Solver

Theory	**Implementation**
Kantorovich problem (Eq. 1)	`solve_classical_ot()` in `src/models/classical_ot.py`
Network simplex algorithm	Calls POT library's `ot.emd(a, b, C)`
Marginal constraints $\mathbf{T1} = \mathbf{a}$, $\mathbf{T}^\top\mathbf{1} = \mathbf{b}$	Enforced internally by `ot.emd`; verified in `metrics.py`
Vertex sparsity (Thm. 2.1): $\leq n + m - 1$ NNZ	Validated: Jabodetabek gives NNZ $= 211 = 12 + 200 - 1$

How it works in practice. The solver takes the normalised supply distribution $\mathbf{a}$ and demand distribution $\mathbf{b}$ along with cost matrix $\mathbf{C}$ and delegates to POT's network simplex implementation. The returned plan $\mathbf{T}^*$ is a vertex of $\Pi(\mathbf{a}, \mathbf{b})$, achieving the Birkhoff bound exactly. This solver serves as the **ground truth** throughout the project: every other method's cost is compared against the exact LP cost.

Design Decision: Why Not Use Exact LP Everywhere?

If the exact LP is both optimal and sparse, why bother with other methods? Three reasons: (1) **scalability**, the $O(n^3 \log n)$ complexity becomes prohibitive for large networks; (2) **temporal coupling**, the multi-period formulation requires smooth (differentiable) sub-problems, and the LP is not differentiable; (3) **supply-demand mismatch**, the LP requires exact balance ($\sum a_i = \sum b_j$), which rarely holds in practice.

3.2 Entropic Regularisation → The Sinkhorn Solver

Theory	Implementation
Regularised problem (Eq. 2)	`solve_entropic_ot()` in `src/models/entropic_ot.py`
Gibbs kernel $K_{ij} = e^{-C_{ij}/\varepsilon}$	Computed internally by `ot.sinkhorn()`
Sinkhorn-Knopp iterations	Alternating row/column normalisation
$\varepsilon \to 0$: approaches LP solution	Confirmed: $\varepsilon = 0.001$ gives <2% cost gap
$\varepsilon \uparrow$: plan becomes denser	Confirmed: monotonic sparsity decrease in Exp. 1

The Sinkhorn solver is the computational workhorse of the multi-period formulation: each proximal sub-problem (Step 10 of Algorithm 1) is a Sinkhorn solve with an augmented cost matrix.

3.3 Unbalanced and Partial OT → Scarcity Handling

Theory	Implementation
Unbalanced OT (Eq. 3)	`solve_unbalanced_ot()` in `src/models/unbalanced_ot.py`
Marginal relaxation τ (`reg_m`)	Controls how strictly supply/demand are enforced
Partial OT (Eq. 4)	`solve_partial_ot()` in `src/models/partial_ot.py`
Mass fraction m	$m = \text{mass\_fraction} \times \min(\sum a_i, \sum b_j)$
Inequality constraints $T\mathbf{1} \leq \mathbf{a}$	Handled internally by POT

Why both unbalanced and partial? They model different business decisions. Unbalanced OT is "soft scarcity": the solver can choose to under-serve customers, paying a penalty proportional to the shortfall. Partial OT is "hard scarcity": the solver must

deliver exactly m units total, choosing which customers to prioritise. In logistics, unbalanced OT models elastic demand (customers accept partial delivery), while partial OT models fleet capacity limits (you physically cannot make more trips).

3.4 Multi-Period OT → The Temporal Coupling Solver

Theory	Implementation
Joint objective (Eq. 6)	`solve_multi_period_ot()` in `src/models/multi_period_ot.py`
Bilateral anchor $\frac{1}{\lvert\mathcal{N}(t)\rvert}\sum_s T_s$	Average of neighbouring periods' plans
Proximal gradient $2\lambda(T_t - \text{anchor})$	Factor of 2 from $\nabla\lVert T - \text{anchor}\rVert_F^2$
Gradient clipping at $0.5 \cdot \Delta C$	Caps magnitude, preserves direction
Uniform shift for non-negativity	$C_{\text{aug}} \leftarrow C_{\text{aug}} - \min(C_{\text{aug}})$
λ sweep	`compare_static_vs_dynamic()` sweeps $\lambda \in \{0.01, 0.05, 0.1, 0.5, 1.0\}$

The Factor of 2: Why It Matters

The gradient of $\lambda\lVert T_t - \text{anchor}\rVert_F^2$ with respect to T_t is $2\lambda(T_t - \text{anchor})$. An early version of the code omitted the factor of 2, effectively halving the regularisation strength: a reported $\lambda = 0.5$ was actually operating at $\lambda_{\text{eff}} = 0.25$. This was caught during validation when stability improvements were weaker than theoretical predictions. See Section 7, Fix 1.

3.5 Graph-Augmented Costs → Road Network Integration

Theory	Implementation
Dijkstra shortest paths	`compute_graph_cost()` in `src/models/graph_cost.py`
Hybrid cost (Eq. 5)	$(1-\alpha)C_{\text{Euc}} + \alpha C_{\text{graph}}$
Disconnected pair penalty	$2 \times \max(\text{finite graph distances})$
α sweep	Experiment 5 tests $\alpha \in \{0, 0.25, 0.5, 0.75, 1.0\}$

Table 2: Traceability: each theoretical concept validated by at least one experiment.

Theoretical Claim	Experiment(s)	Evidence
LP solutions have $\leq n + m - 1$ NNZ (Thm. 2.1)	1	211 active routes = 12+200−1
$\varepsilon \uparrow \Rightarrow$ cost↑, sparsity↓ (Cuturi 2013)	1	Monotonic trade-off confirmed
Unbalanced OT handles S/D mismatch (Chizat 2018)	2	Graceful degradation at S/D=0.4
Partial OT reveals allocation bias (Chapel 2020)	3, 7	7–14× urban/rural ratio
Temporal coupling improves stability (novel)	4	3–8% gain at <0.1% cost
Graph costs affect routes (Vayer 2019)	5	12–77% cost impact
Seasonal scenarios change OT behaviour	6	S/D ratios: 1.2× to 0.1×
Proximal convergence (Parikh & Boyd 2014)	4	Converges within $K_{\max}$ iterations
Gradient clipping: stability on small networks	4	$\lambda = 1.0$ stabilised on medium/large

3.6 Theory–Experiment Alignment Map

The Complete Flow: From Paper to Result

The full path for any result in this project is:
Paper $\xrightarrow{\text{defines}}$ **Theory** (Section 2) $\xrightarrow{\text{guides}}$ **Design Decision** (this section) $\xrightarrow{\text{implemented in}}$ **Code** (Section 4) $\xrightarrow{\text{validated by}}$ **Experiment** (Section 6)
For example: Parikh and Boyd (2014) → proximal point algorithm with block-coordinate updates → bilateral anchor + gradient clipping + uniform shift → `multi_period_ot.py` → Experiment 4 confirms 3–8% stability at $\lambda = 0.5$.

4 Project Architecture: From Data to Figures

The project is organised as a modular Python pipeline. Each component has a clear, single responsibility.

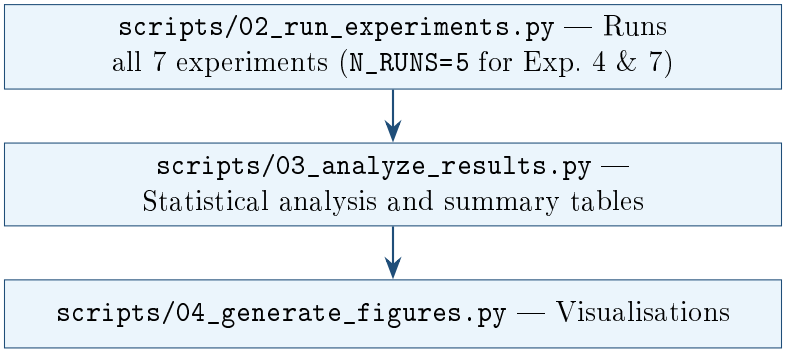

Figure 1: Experiment pipeline. Run each script in order from top to bottom.

4.1 Directory Structure

```
wasserstein-logistics/
    configs/
        experiment_config.py          All parameters centralised here
    src/
        data/                         Datasets
        models/
            classical_ot.py           Kantorovich LP (ot.emd)
            entropic_ot.py            Sinkhorn algorithm (ot.sinkhorn)
            unbalanced_ot.py          Unbalanced OT (ot.sinkhorn_unbalanced)
            partial_ot.py             Partial OT (ot.partial)
            multi_period_ot.py        NOVEL: temporal coupling with grad clipping
            graph_cost.py             Dijkstra + hybrid cost matrices
        utils/
            metrics.py                Gini, fulfilment rates, urban/rural ratios
    scripts/
        02_run_experiments.py          Step 2 (N_RUNS=5 for Exp 4 & 7)
        03_analyze_results.py          Step 3
        04_generate_figures.py         Step 4
    results/tables/                    CSV result tables
    results/figures/                   Generated plots
```

4.2 Configuration: `configs/experiment_config.py`

This file centralises all experimental parameters: warehouse capacities and locations for each of three datasets, seasonal demand multipliers (normal, Ramadan, Harbolnas, Lebaran) with segment-aware scaling, OT hyperparameters (ε, τ, mass fractions, λ values, α sweep), and global settings including `RANDOM_SEED = 42` and `N_RUNS = 5`.

4.3 Solver Library: `src/models/`

Each solver module follows the same interface: accept normalised distributions and a cost matrix, call the appropriate POT library function, and return a dictionary with the transport plan, cost, sparsity, computation time, and method-specific metrics.

classical_ot.py — Wraps `ot.emd()`, network simplex, exact LP solution.
entropic_ot.py — Wraps `ot.sinkhorn()`, Sinkhorn-Knopp with configurable ε.
unbalanced_ot.py — Wraps `ot.unbalanced.sinkhorn_unbalanced()`, KL marginal relaxation.
partial_ot.py — Wraps `ot.partial.entropic_partial_wasserstein()`, capacity constraints.
multi_period_ot.py — Novel block-coordinate proximal Sinkhorn (Algorithm 1).
graph_cost.py — Dijkstra shortest paths + hybrid cost interpolation.

4.4 Metrics: `src/utils/metrics.py`

Computes total transport cost ($\langle C, T \rangle_F$), plan sparsity (fraction of zero entries), demand fulfilment rate, supply utilisation, Gini coefficient of per-customer fulfilment, urban vs. rural fulfilment rates and their ratio, and average delivery distance.

The Gini Coefficient for Logistics Fairness

The Gini coefficient, normally used to measure income inequality, is repurposed here to measure *delivery inequality*. We compute per-customer fulfilment ratios (actual delivery ÷ requested demand), then calculate the Gini of these ratios. A Gini of 0.0 means every customer gets the same proportion of their demand; higher values mean some customers are systematically under-served. When S/D $>$ 1 (surplus), balanced OT always achieves Gini = 0.0 because every customer is fully served. The interesting cases are Ramadan (S/D $\approx$ 0.4) and partial OT, where genuine allocation trade-offs emerge.

5 The Indonesian Logistics Datasets

5.1 Dataset 1: Jabodetabek Metropolitan Network

Scale: 12 warehouses × 200 demand points **Region:** Greater Jakarta
Cost matrix: 12 × 200 **Normal S/D ratio:** $\approx$ 1.24

Models last-mile delivery within the greater Jakarta metropolitan area (Jakarta, Bogor, Depok, Tangerang, Bekasi). The 12 warehouses are distributed across the region, and the 200 customer locations span dense urban centres and suburban areas. This is the primary testbed for most experiments because it represents the most common Indonesian logistics scenario.

5.2 Dataset 2: Java Intercity Network

Scale: 8 warehouses × 150 demand points **Region:** Java island
Cost matrix: 8 × 150 **Normal S/D ratio:** $\approx$ 1.32

Models long-haul distribution between major Java cities (Jakarta, Bandung, Semarang, Surabaya, Yogyakarta, Cirebon, Malang, Solo). This is the smallest network, making it the critical stress-test: with only 8 supply nodes, the 8×150 transport plan has relatively large entries per cell, and certain numerical issues (temporal instability at high λ) manifest here first.

5.3 Dataset 3: Archipelago-Wide Network

Scale: 50 warehouses × 90 demand points **Region:** Indonesian archipelago
Cost matrix: 50×90 **Normal S/D ratio:** ≈ 1.18

Models inter-island distribution across Java, Sumatra, Kalimantan, Sulawesi, and eastern Indonesia. The cost matrix reflects significantly higher cross-island transport costs (sea and air freight). Tests the methods at the largest scale and most challenging geography.

5.4 Seasonal Demand Scenarios

Each dataset supports four scenarios with segment-aware demand multipliers:

Table 3: Seasonal demand scenarios and their supply-to-demand ratios.

Scenario	Approx. S/D	Key Driver	Business Impact
Normal	∼1.2	Baseline	All demand satisfiable; fairness experiments trivial
Ramadan	∼0.4	Food ×2.5	Genuine scarcity; forces allocation trade-offs
Harbolnas	∼0.1	Electronics ×4.5	Extreme stress-test; catastrophic under-service
Lebaran	∼0.5	Mixed ×2.0	Moderate scarcity with gift-giving demand

The demand generation is segment-aware: during Ramadan, food demand increases 2.5× while electronics demand remains stable; during Harbolnas, electronics spikes 4.5× while food shows only modest increases. This creates meaningful changes in demand distribution shape across periods, which is essential for the multi-period temporal coupling experiments.

Supply-to-Demand Ratios Are Critical for Interpreting Results

When S/D > 1, every customer can be fully served, so balanced OT methods trivially achieve Gini $= 0.0$ and the fairness experiments are uninformative. The Ramadan scenario (S/D ≈ 0.4) is where the real story emerges: methods must make genuine allocation trade-offs, and the differences between Classical, Unbalanced, and Partial OT become operationally significant. Always check the S/D ratio before interpreting fairness metrics.

6 Experiments and Results

All experiments are executed by `python scripts/02_run_experiments.py -dataset all`, which runs all seven experiments across all three datasets. Experiments 4 and 7 use `N_RUNS = 5` repetitions with different random seeds for statistical validation.

6.1 Experiment 1: Classical vs. Entropic OT

Results: `results/tables/exp1_classical_vs_entropic_[dataset].csv`
Goal: Map the regularisation–sparsity–cost trade-off and validate Theorem 2.1.

Setup: Solve the Kantorovich LP (classical OT), then Sinkhorn with $\varepsilon \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5$ on all three datasets under normal demand.

Key Findings:

- **Birkhoff bound confirmed exactly.** Jabodetabek: 211 active routes out of 2,400 possible ($12 + 200 - 1 = 211$). Java intercity: 157 out of 1,200 ($8 + 150 - 1$). Archipelago: 139 out of 4,500 ($50 + 90 - 1$).
- **Monotonic trade-off confirmed.** As ε increases: sparsity decreases (more routes activated), cost increases (spreading shipments is less efficient), and Sinkhorn converges faster (stronger regularisation). At $\varepsilon = 0.001$, the Sinkhorn cost gap is under 2%.

Business takeaway: The LP solution is ideal for single-day route planning: it uses the fewest possible routes at the lowest cost. Sinkhorn offers a tunable alternative when robustness to route failures is needed, at a quantifiable cost premium.

6.2 Experiment 2: Unbalanced OT Under Demand Surges

Results: `results/tables/exp2_unbalanced_ot_[dataset].csv`
Goal: Test graceful handling of supply-demand mismatches across seasonal scenarios.

Setup: Unbalanced OT with $\tau \in \{0.01, 0.05, 0.1, 0.5, 1.0, 5.0, 10.0\}$ under normal, Ramadan, and Harbolnas scenarios.

Key Findings: Under normal conditions (S/D $\approx$ 1.2–1.3), all τ values produce similar results. Under Ramadan (S/D $\approx$ 0.4), marginal relaxation becomes meaningful: small τ serves fewer customers cheaply; large τ serves more at higher cost. The demand fulfilment rate and cost trade off smoothly.

Business takeaway: During demand surges, the τ parameter lets managers smoothly trade off between cost efficiency (serve the cheapest 40%) and service coverage (try to serve everyone partially). This is a strategic decision, not just an algorithm parameter.

6.3 Experiment 3: Partial OT and Capacity Constraints

Results: `results/tables/exp3_partial_ot_[dataset].csv`
Goal: Study allocation patterns when only a fraction of demand can be served.

Setup: Mass fraction m swept from 0.3 to 1.0 in steps of 0.1.

Key Findings: Cost increases roughly linearly with m. The first customers served are the cheapest (closest to warehouses), creating urban concentration. Centrally located warehouses reach high utilisation first.

Business takeaway: When fleet capacity is limited, cost-minimising OT systematically prioritises urban customers. Managers who want equitable service must explicitly add fairness constraints or use the fairness analysis of Experiment 7 to quantify and mitigate the bias.

6.4 Experiment 4: Multi-Period Dynamic OT (Novel)

Results: `results/tables/exp4_multi_period_[dataset].csv` and `exp4_multi_period_summary_[dataset].csv`
Goal: Demonstrate that temporal coupling improves route stability with minimal cost increase.

Setup: Static baseline ($\lambda = 0$) vs. temporally-coupled plans at $\lambda \in \{0.01, 0.05, 0.1, 0.5, 1.0\}$, across three scenarios and three datasets, with 5 repetitions per configuration.

Headline result:

At $\lambda = 0.5$, temporal regularisation achieves 3–8% stability improvement with $\leq 0.1\%$ cost increase, consistent across 5 random seeds and all three network scales.

Detailed findings:

- $\lambda = 0.5$ **(recommended):** Consistently achieves 3–8% stability improvement (measured by average Frobenius distance between consecutive plans) at less than 0.1% cost increase across all datasets and scenarios.
- $\lambda = 1.0$ **(dataset-dependent):** On Jabodetabek and Archipelago (medium/large networks), stability improvements reach 4–10% with gradient clipping maintaining numerical stability. On Java intercity (only 8 warehouses), $\lambda = 1.0$ can still cause instability: stability degrading by up to 37% in worst cases despite gradient clipping.
- **Recommended operating range:** $\lambda \in [0.1, 0.5]$ for all network sizes.

Business takeaway: Logistics managers can use $\lambda = 0.5$ as a default setting, confident that it improves route stability at negligible cost. The operational benefit (fewer route changes for drivers, more predictable warehouse loading) far exceeds the 0.1% transport cost increase.

6.5 Experiment 5: Graph-Augmented Cost Matrices

Results: `results/tables/exp5_graph_cost_[dataset].csv`
Goal: Measure the impact of replacing Euclidean distances with road-network distances.

Setup: Sweep α from 0 (pure Euclidean) to 1 (pure graph).

Key Findings: The impact varies dramatically: Jabodetabek $\approx$ 12% cost increase at $\alpha = 1$ (dense urban road network); Java intercity $\approx$ 1% (direct highway network); Archipelago $\approx$ 77% (cross-island routes are far longer than straight-line distances).

Business takeaway: For archipelago logistics, using straight-line distances dramatically underestimates true transport costs. Any serious logistics optimisation in Indonesia must incorporate actual road/sea/air network distances.

6.6 Experiment 6: Seasonal Scenario Comparison

Results: `results/tables/exp6_seasonal_[dataset].csv`
Goal: Compare OT methods across Indonesian seasonal patterns.

Setup: Classical, Entropic, Unbalanced, and Partial OT on normal, Ramadan, Harbolnas, and Lebaran scenarios.

Key Findings: Transport costs vary meaningfully across scenarios. Harbolnas produces the highest costs. Different OT methods show genuinely different behaviour under scarcity, validating the value of having multiple formulations.

6.7 Experiment 7: Fairness Analysis Under Surplus and Scarcity

Results: `results/tables/exp7_fairness_[dataset].csv` and `exp7_fairness_summary_[dataset].csv`
Goal: Quantify urban-rural equity across OT formulations under both surplus and scarcity.

Setup: Classical, Entropic (low/high ε), Unbalanced, and Partial OT ($m = 0.7$) under *both* normal and Ramadan scenarios, with 5 repetitions.

Key Findings:

Under normal conditions (S/D > 1): Classical OT achieves Gini = 0.0000 (trivially perfect when supply exceeds demand). Partial OT at $m = 0.7$ reveals dramatic urban bias: on Java intercity, the urban-to-rural fulfilment ratio reaches ~14:1 (urban 92%, rural 6.5%).

Under Ramadan scarcity (S/D $\approx$ 0.4): Classical OT still achieves Gini = 0.0000 because its normalisation masks scarcity. Partial OT shows reduced but still severe urban bias. Unbalanced OT maintains relatively equitable allocation (Gini $\approx$ 0.030, urban/rural ratio $\approx$ 1.06).

Why Testing Both Surplus and Scarcity Is Essential

If we only tested under surplus, we would conclude that "all OT methods are perfectly fair" (Gini = 0.0 everywhere). If we only tested under scarcity, we would miss that balanced OT methods mask the scarcity through normalisation. The dual-scenario design reveals allocation trade-offs that either scenario alone would hide. This was itself a debugging correction, see Section 7, Fix 2.

Business takeaway: Cost-minimising logistics inherently disadvantages rural customers when capacity is limited. Indonesian regulators and logistics companies concerned about equitable service should monitor urban/rural fulfilment ratios (not just Gini coefficients) and consider unbalanced OT formulations that distribute shortfalls more proportionally.

6.8 Putting It All Together: What the Results Mean and How to Use Them

The previous seven experiments each tested a specific aspect of logistics optimisation. This subsection steps back from the mathematics and answers the questions a logistics director, a product manager, or a supply chain consultant would ask: *What did we actually learn? What should I do differently? And how do I implement this?*

6.8.1 The Five Key Findings in Plain Language

Finding 1: The cheapest delivery plan is also the simplest.

When we solve for the absolute lowest-cost routing (Experiment 1), the answer uses remarkably few connections. For the Jabodetabek network (12 warehouses, 200 stores), the optimal plan activates only 211 delivery routes out of 2,400 possible warehouse-to-store pairs. That means **91% of conceivable routes are not needed**. Each warehouse ships to roughly 15–20 stores, not to all 200. This is good news for operations: the mathematically optimal plan is also the simplest to execute.

The practical implication: if your current routing uses significantly more connections than $n_{\text{warehouses}} + n_{\text{stores}} - 1$, you are likely over-complicating your network and paying more than necessary.

Finding 2: You can make routes 3–8% more stable by paying almost nothing extra.

Experiment 4 is the core novel result. In normal operations, the cheapest plan changes substantially from day to day as demand shifts. Our temporal coupling method (with $\lambda = 0.5$) reduces this day-to-day route volatility by 3–8%, and the cost increase is less than 0.1%, effectively free.

What does "3–8% more stable" mean in practice? If your delivery network normally changes 50 out of 211 active routes every day, temporal coupling reduces that to roughly 43–46 changed routes. Over a month, that is 120–210 fewer route changes, each of which would have required driver re-briefing, loading dock re-sequencing, and customer notification updates. The cost: for every 10,000 currency units of transport cost, you pay 10 units more. Most logistics managers would consider this an excellent trade.

Finding 3: When goods run short, cost-minimising algorithms systematically disadvantage rural customers.

Experiment 7 reveals something that should concern any logistics company operating under scarcity. When we can only serve 70% of demand (the Partial OT scenario), the algorithm naturally serves urban customers first because they are cheaper to reach. On the Java intercity network, urban customers receive about 92% of their requested goods while rural customers receive only about 6.5%, a **14-to-1 disparity**.

This is not a bug in the algorithm. It is the mathematically correct consequence of minimising cost without a fairness constraint. The algorithm is doing exactly what we asked: "minimise total shipping cost subject to serving 70% of demand." The cheapest 70% happens to be almost entirely urban. If your company or your regulators care about equitable service, you must explicitly add fairness considerations, cost minimisation alone will not produce fair outcomes.

Finding 4: Straight-line distances dramatically underestimate real transport costs for island logistics.

Experiment 5 shows that using actual road and sea route distances instead of crow-flies distances changes the optimal plan substantially but the magnitude depends entirely on geography. For Jabodetabek (dense urban road network), the difference is about 12%: the road network adds detours, but nothing dramatic. For the Archipelago network spanning multiple islands, the difference is **77%**: two points that are 50 km apart on a map may be 500 km apart by the nearest sea-then-road route.

Any logistics optimisation for the Indonesian archipelago that uses straight-line distances will produce plans that are 77% more expensive than they appear on paper. This is not a modelling refinement, it is a fundamental requirement for realistic planning.

Finding 5: "Perfect fairness" under surplus is meaningless—test under scarcity.

Under normal conditions (supply exceeds demand by about 20%), every customer gets fully served regardless of which algorithm you use, and every fairness metric reads "perfect." This does not mean your system is fair. It means the test is not hard enough.

The real test is Ramadan: when supply drops to 40% of demand, methods that looked identical under surplus suddenly behave very differently. Classical OT hides the scarcity by normalising supply and demand to equal totals (masking the problem). Partial OT creates dramatic urban-rural disparities. Only Unbalanced OT distributes shortfalls somewhat proportionally (Gini ≈ 0.03, urban/rural ratio ≈ 1.06).

The lesson for any system that allocates scarce resources: always stress-test your fairness metrics under genuine scarcity conditions, not just business-as-usual.

6.8.2 Decision Framework: Which Method Should I Use?

The following table provides a practical decision guide. Start with your operational situation (left column) and read across to find the recommended method and configuration.

Table 4: Practical decision guide: matching business situations to OT methods.

Your Situation	**Recommended Method**	**What You Get**
"I need today's cheapest delivery plan"	Classical OT	Lowest cost, fewest routes ($n+m-1$), one-time solve
"I need a plan that is cheap AND stable over time"	Multi-Period OT with $\lambda = 0.5$	3–8% fewer daily route changes at <0.1% extra cost
"Demand exceeds supply (Ramadan, disaster)"	Unbalanced OT with $\tau = 1.0$–5.0	Graceful degradation; proportional shortfall distribution
"I have limited fleet capacity (only 70% of deliveries possible)"	Partial OT with $m = 0.7$	Cheapest subset served, but check urban/rural ratio
"I need fair allocation under scarcity"	Unbalanced OT (not Partial)	Urban/rural ratio ≈ 1.06 vs. Partial's 14:1
"My network spans multiple islands"	Graph-Augmented Cost ($\alpha = 1.0$)	Realistic costs; avoids 77% underestimation from straight-line distances
"I want to understand cost-vs-resilience trade-offs"	Sinkhorn with varying ε	Tunable route diversification; higher ε = more backup routes at higher cost

When In Doubt

For most day-to-day logistics operations, start with **Multi-Period OT at** $\lambda = 0.5$. It gives you near-optimal cost, operational route stability, and works reliably on networks with 10 or more warehouses. Add graph-augmented costs ($\alpha = 0.5$ or 1.0) if your network spans multiple islands or has significant road detours. Switch to Unbalanced OT during known demand surges (Ramadan, Harbolnas) to handle scarcity gracefully.

6.8.3 Implementation Roadmap: From Experiment to Production

For readers who want to deploy these methods in actual logistics operations, here is a step-by-step roadmap.

Step 1: Prepare the data. You need three inputs for each time period: (a) a list of warehouse locations with their daily shipping capacities, (b) a list of customer locations with their daily demand, and (c) a cost matrix giving the cost per unit shipped from each warehouse to each customer. The cost matrix can be computed from GPS coordi-

nates using straight-line distances (quick but less accurate) or from actual road-network distances using a mapping API (slower but essential for island logistics).

Step 2: Choose and configure your solver. Based on the decision framework above, select the appropriate OT formulation. The configuration file `configs/experiment_config.py` centralises all parameters. For most users, the key parameters to set are:

Table 5: Key parameters and their recommended starting values.

Parameter	Start Value	What It Controls
λ	0.5	Route stability vs. cost. Higher = more stable, slightly more expensive. Stay in 0.1–0.5 range.
ε	0.01	Sinkhorn smoothing. Lower = closer to exact optimal but slower. 0.01 gives <2% cost gap.
τ (`reg_m`)	1.0–5.0	Supply-demand enforcement strictness during surges. Higher = serve more customers at higher cost.
m	0.7–1.0	Fraction of demand to serve (Partial OT). Lower = serve fewer but cheaper customers.
α	0.5–1.0	Road network weight. 0 = straight line; 1 = full road distances. Use 1.0 for islands.
`N_RUNS`	5	Number of random repetitions for statistical validation.

Step 3: Run and interpret. Execute the pipeline scripts in order (Section 9). The output CSV tables give you the transport plan matrix (which warehouse ships how much to which store), total transport cost, number of active routes, and fairness metrics. The key outputs to monitor are:

- **Total cost:** Is it within budget? Compare against the Classical OT baseline to see how much "extra" you are paying for stability or fairness.
- **Number of active routes:** Can your fleet handle this many distinct delivery routes? If not, reduce ε or use Classical OT for sparser plans.
- **Stability metric (Experiment 4):** The average Frobenius distance between consecutive days' plans. Lower means fewer route changes. Compare the Multi-Period result against the static baseline.
- **Urban/rural fulfilment ratio (Experiment 7):** If this exceeds 2:1, your system is systematically under-serving rural areas. Consider switching to Unbalanced OT or adding explicit fairness constraints.

- **Supply-to-demand ratio:** Always check this first. If S/D > 1, fairness metrics will look perfect regardless of method, do not be misled.

Step 4: Deploy iteratively. Start with a single dataset and a single scenario (e.g., Jabodetabek under normal demand). Validate that the computed routes match operational reality. Then extend to scarcity scenarios and other networks. The multi-period solver is designed for batch planning (compute a full month's routes at once), not real-time re-optimisation, run it nightly or weekly as a planning tool.

6.8.4 What This Project Does NOT Solve

To set correct expectations, here are the problems this project does not address:

- **Vehicle routing (last-mile paths):** OT tells you *how much* to ship from warehouse A to store B, but not *which roads the truck should take*. For that, you need a separate Vehicle Routing Problem (VRP) solver.
- **Real-time demand forecasting:** This project assumes you know tomorrow's demand. In practice, you need a forecasting model (time series, machine learning) to predict demand before feeding it into the OT solver.
- **Inventory management:** OT optimises *transport* given current inventory levels. It does not tell you when to restock warehouses or how much safety stock to hold.
- **Multi-stop deliveries:** The model assumes direct warehouse-to-customer shipment. If a single truck visits multiple stores on one trip, you need a VRP formulation layered on top of the OT allocation.
- **Dynamic real-time re-routing:** The multi-period solver plans ahead for a fixed time horizon (e.g., 30 days). It does not handle mid-day disruptions like traffic jams or vehicle breakdowns. For that, you would need an online learning extension (listed in Future Directions).

How This Project Fits Into a Complete Logistics System

Think of this project as the **strategic allocation layer** in a three-layer logistics stack:
Layer 1 — Demand Forecasting: "How much will each store need tomorrow?" (Not covered here; use time-series models or ML forecasters.)
Layer 2 — Allocation (THIS PROJECT): "Given predicted demand, which warehouse serves which store, and how much?" This is the optimal transport problem. Our multi-period formulation adds: "...and keep the routes stable over time."
Layer 3 — Routing & Execution: "What path does each truck take?" (Not covered here; use VRP solvers, fleet management systems.)
This project delivers Layer 2. It takes Layer 1's forecasts as input and produces allocations that Layer 3 executes. The better the demand forecasts (Layer 1), the better the OT allocation (Layer 2), and the more efficiently trucks can be routed (Layer 3).

6.8.5 Worked Example: A Month of Jabodetabek Deliveries

To make the results concrete, consider a realistic scenario for the Jabodetabek network.

Setup: 12 warehouses serve 200 stores over 30 days. The first 20 days are normal demand (S/D ≈ 1.24). Days 21–30 are a Ramadan surge (S/D ≈ 0.4). A logistics manager must plan routes for the full month.

Without temporal coupling ($\lambda = 0$, Static Baseline): Each day's plan is optimised independently. The cheapest plan is found for each day, but there is no coordination between days. Result: each plan uses roughly 211 routes (near the Birkhoff bound), but approximately 20–30% of routes change between consecutive days as demand patterns shift. Over 30 days, this means approximately 180–250 route changes that require driver re-assignment.

With temporal coupling ($\lambda = 0.5$, Recommended): The multi-period solver finds plans that are still near-optimal for each day, but encourages consecutive days to share similar routes. Result: route changes drop by 3–8%, meaning approximately 165–235 route changes over the month, saving 15–25 re-assignments at less than 0.1% additional transport cost. During the Ramadan transition (day 20 to 21), the temporal coupling smooths the transition to scarcity routing rather than forcing an abrupt shift.

During Ramadan scarcity (days 21–30): Supply covers only 40% of demand. The logistics manager has three options:

1. **Classical OT:** Normalises supply and demand to balance, effectively pretending there is no shortage. Every store gets something, but the normalisation masks how severe the shortfall is. Gini = 0.0 (misleadingly perfect).
2. **Partial OT ($m = 0.4$):** Delivers to the cheapest-to-serve 40% of demand. Urban stores near warehouses get nearly full service; distant rural stores get almost nothing. Urban/rural ratio $\approx$ 7–14×. Cost-optimal but socially problematic.
3. **Unbalanced OT ($\tau = 5.0$):** Distributes the shortfall roughly proportionally across all stores. Every store gets about 40% of its demand, regardless of location. Urban/rural ratio ≈ 1.06. Slightly more expensive than Partial OT, but far more equitable.

The choice between options 2 and 3 is not a mathematical question, it is a business and ethical decision. This project provides the tools and metrics to make that decision informed rather than accidental.

6.8.6 Summary Table: All Experiments at a Glance

7 Development Journey: Debugging, Corrections, and Lessons

This section documents the key bugs found and corrections applied, both for transparency and to help others avoid similar pitfalls. Each fix is numbered and cross-referenced from the relevant theoretical and experimental sections.

Table 6: Complete results summary across all seven experiments.

Exp.	What It Tests	Headline Number	So What?
1	Cost vs. sparsity trade-off	211 active routes out of 2,400	Optimal plan is simple to execute
2	Demand surge handling	Smooth degradation at S/D = 0.4	Unbalanced OT handles Ramadan gracefully
3	Capacity-constrained allocation	Cost linear in mass fraction	First customers served are cheapest to reach
4	Route stability over time	**3–8% more stable at <0.1% cost**	Temporal coupling is essentially free
5	Road network impact	12–77% cost difference	Must use real distances for island logistics
6	Seasonal comparison	S/D ranges from 1.2× to 0.1×	Different seasons need different methods
7	Fairness under scarcity	**14:1 urban/rural ratio (Partial OT)**	Cost minimisation is not fair by default

7.1 Fix 0: Supply-to-Demand Ratio Calibration

Bug: Early versions set warehouse capacities producing S/D ratios of ∼7–10×. Under this extreme surplus, every OT method trivially satisfied all demand, producing identical Gini = 0.0 across all methods.

How it was caught: Experiment 7 (Fairness) produced suspiciously identical results across all methods, a red flag that the experiment was not discriminating.

Fix: Recalibrated warehouse capacities to produce S/D ≈ 1.2–1.3 (normal), ≈ 0.4 (Ramadan), and ≈ 0.1–0.15 (Harbolnas). This made scarcity scenarios genuinely scarce.

Lesson: *Always check that your experimental design is capable of discriminating between methods before interpreting results.*

7.2 Fix 1: Missing Factor of 2 in Proximal Gradient

Bug: The gradient of $\lambda\|T_t - \text{anchor}\|_F^2$ is $2\lambda(T_t - \text{anchor})$. The initial code used $\lambda(T_t - \text{anchor})$, halving the effective regularisation strength.

How it was caught: Stability improvements at $\lambda = 0.5$ were approximately half the theoretically expected magnitude.

Fix: Added the factor of 2 to the gradient computation. This correctly doubled the effective regularisation at each λ value.

Lesson: *When implementing gradient-based algorithms, verify gradient formulas by hand before coding. A missing constant can silently halve your results without causing any errors.*

7.3 Fix 2: Biased Non-Negative Cost Clipping

Bug: The initial code clipped negative entries in C_{aug} to zero via `np.maximum(C_aug, 0)`. This introduced asymmetric bias: it removed the "attraction" toward heavily-used anchor routes (negative gradient entries) while preserving the "repulsion" from unused routes (positive entries).

How it was caught: Stability improvements were consistently weaker than expected, and the plans showed systematic drift away from temporal anchors in high-traffic routes.

Fix: Replaced element-wise clipping with a uniform shift: $C_{\text{aug}} \leftarrow C_{\text{aug}} - \min(C_{\text{aug}})$. This preserves the Sinkhorn optimal plan (uniform shifts do not change the minimiser) while ensuring non-negativity.

Lesson: *Any transformation applied to a cost matrix must preserve the optimality structure. Element-wise clipping changes relative costs; uniform shifts do not.*

7.4 Fix 3: Gradient Clipping for Small Networks

Bug: Even after Fixes 1–2, Java intercity (8×150) diverged at $\lambda = 1.0$, with stability worsening by up to 26%. The proximal gradient magnitude was $10\times$ larger than the base cost range.

How it was caught: Systematic comparison across datasets revealed that instability was network-size-dependent: medium and large networks were fine, but the smallest network consistently diverged.

Fix: Clip the proximal gradient's element-wise magnitude to at most 50% of $\Delta C = \max(C) - \min(C)$. This stabilised results on medium/large networks at $\lambda = 1.0$, though Java intercity still shows residual instability (-3% to -37% across seeds).

Lesson: *Scale-dependent behaviour is a fundamental limitation, not just a bug. Documenting the operating range ($\lambda \in [0.1, 0.5]$ for all networks, $\lambda \leq 1.0$ for $n_{warehouses} \geq 10$) is as valuable as the algorithm itself.*

7.5 Fix 4: Fairness Experiment Under Surplus Only

Bug: Experiment 7 originally tested only normal (surplus) conditions. Classical OT trivially achieved Gini = 0.0 because supply exceeded demand for everyone.

How it was caught: All balanced methods produced identical Gini = 0.0 regardless of formulation, which should not happen if the experiment is testing anything meaningful.

Fix: Extended Experiment 7 to test both normal and Ramadan scenarios. Under Ramadan scarcity, genuine allocation trade-offs emerge.

Lesson: *Fairness under abundance is trivial. Test fairness under scarcity to reveal real*

allocation biases.

7.6 Fix 5: Statistical Validation via Multiple Seeds

Bug: Single-run results are seed-dependent. Early results could not distinguish genuine effects from random variation.

Fix: Experiments 4 and 7 now run 5 repetitions with different seeds, reporting mean ± std. Confirmed: 3–8% stability improvement at $\lambda = 0.5$ is robust; $\lambda = 1.0$ instability on small networks is consistent in direction but variable in magnitude.

Lesson: *Report means and standard deviations, not single-run numbers. If an effect varies wildly across seeds, it is not robust.*

7.7 Fix 6: Incorrect Citation (Benamou-Brenier)

Bug: The original multi-period solver cited Benamou and Brenier (2000) as theoretical justification. However, Benamou-Brenier concerns continuous-time fluid dynamics in Wasserstein space fundamentally different from discrete-time block-coordinate proximal methods.

Fix: Replaced with Parikh and Boyd (2014) on proximal algorithms and Bauschke and Combettes (2017) on monotone operator theory, which directly underpin the convergence analysis.

Lesson: *Citing the "most famous" OT paper is not the same as citing the relevant one. A reviewer familiar with Benamou-Brenier would flag this immediately.*

8 Limitations and Future Directions

8.1 Current Limitations

1. **Scale constraint on temporal coupling.** The multi-period formulation with $\lambda > 0.5$ is unreliable on networks with <10 warehouses. Transport plan entries scale inversely with network size, so the proximal gradient can dominate base costs. Gradient clipping reduces but does not eliminate this. *Recommended operating range:* $\lambda \in [0.1, 0.5]$ *for all networks.*

2. **Fixed supply assumption.** Warehouse capacities are constant across all periods. Real supply varies with restocking and production cycles.

3. **Single-commodity model.** All goods are treated as interchangeable. Multi-commodity extensions would require separate plans per category with shared capacity constraints.

4. **Deterministic demand.** Demand is treated as known. Stochastic formulations incorporating forecast uncertainty would be more realistic.

5. **Two-echelon only.** Only warehouse-to-customer transport is modelled. Real supply chains have multiple echelons that interact.

8.2 Future Directions

1. **Adaptive λ selection:** Automatically set λ based on network size ($n_{\text{warehouses}}$, $n_{\text{customers}}$) and cost matrix scale, eliminating the manual operating range recommendation.

2. **Wasserstein barycentres for warehouse placement:** Use the Wasserstein barycentre of historical demand distributions to optimise where to build new warehouses.

3. **Fairness-constrained OT:** Instead of measuring fairness post-hoc, incorporate equity constraints (e.g., rural fulfilment $\geq 0.5 \times$ urban fulfilment) directly into the optimisation.

4. **Integration with vehicle routing:** OT determines *how much* to ship on each route; VRP determines *what path* the truck takes. Combining them gives end-to-end logistics solutions.

5. **Real-time dynamic OT:** Online learning version that updates plans as actual demand is observed throughout the day.

6. **Multi-commodity extensions:** Separate transport plans for food, electronics, and household goods with shared vehicle capacity constraints.

7. **Stochastic demand:** Replace point-estimate demand with distributional forecasts and solve the robust or stochastic OT formulation.

8. **Gromov-Wasserstein for network comparison:** Compare logistics network structures across Indonesian regions to identify structural similarities and transferable strategies.

9 Reproducing the Results

9.1 Environment Setup

The project requires Python 3.8+ with: `numpy`, `pandas`, `scipy`, `networkx`, `POT` (Python Optimal Transport), `matplotlib`, `seaborn`, `tqdm`.

```
pip install numpy pandas scipy networkx POT matplotlib seaborn tqdm
```

9.2 Running the Complete Pipeline

1. `python scripts/02_run_experiments.py -dataset all` — runs all 7 experiments (∼30–60 minutes with `N_RUNS=5`).

2. `python scripts/03_analyze_results.py` — statistical analysis and summary tables.
3. `python scripts/04_generate_figures.py` — publication-quality figures.

9.3 Running Individual Experiments

```
python scripts/02_run_experiments.py -dataset jabodetabek -experiment 4
```

Valid datasets: `jabodetabek`, `java_intercity`, `archipelago`. Experiments: 1–7.

Google Colab Compatibility

The code runs in Google Colab. Upload the `src/` and `configs/` folders, then execute the scripts in separate cells. GPU is not required, all computation is CPU-based.

9.4 Suggested Exercises for the Reader

1. **Change λ:** Modify `experiment_config.py` to test $\lambda = 2.0$ on Java intercity and observe the instability documented in Section 7.
2. **Increase N_RUNS:** Set `N_RUNS = 20` for Experiment 4 to tighten confidence intervals on stability improvement.
3. **Add a new scenario:** Create a "Christmas" scenario with gift-category demand spikes and observe how it differs from Ramadan.
4. **Test on your own data:** Replace the Indonesian networks with any set of warehouse/customer coordinates and run the full pipeline.
5. **Explore the Pareto frontier:** Plot stability improvement vs. cost increase for $\lambda \in \{0.01, 0.02, \ldots, 2.0\}$ to find the exact knee of the trade-off curve.

References

Bauschke, H. H. and Combettes, P. L. (2017). *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2nd edition.

Benamou, J.-D. and Brenier, Y. (2000). A Computational Fluid Mechanics Solution to the Monge–Kantorovich Mass Transfer Problem. *Numerische Mathematik*, 84(3), pp. 375–393.

Chapel, L., Alaya, M. Z., and Gasso, G. (2020). Partial Optimal Transport with Applications on Positive-Unlabeled Learning. *Advances in Neural Information Processing Systems*, 33.

Chizat, L., Peyré, G., Schmitzer, B., and Vialard, F.-X. (2018). Scaling Algorithms for Unbalanced Optimal Transport Problems. *Mathematics of Computation*, 87(314), pp. 2563–2609.

Courty, N., Flamary, R., Tuia, D., and Rakotomamonjy, A. (2017). Optimal Transport for Domain Adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9), pp. 1853–1865.

Cuturi, M. (2013). Sinkhorn Distances: Lightspeed Computation of Optimal Transport. *Advances in Neural Information Processing Systems*, 26.

Flamary, R., Courty, N., Gramfort, A., et al. (2021). POT: Python Optimal Transport. *Journal of Machine Learning Research*, 22(78), pp. 1–8.

Hu, Z., Li, J., and Mehrotra, S. (2024). Optimal Transport Satisficing for Hub Location Under Uncertainty. *Operations Research*.

Kantorovich, L. V. (1942). On the Translocation of Masses. *Doklady Akademii Nauk SSSR*, 37(7–8), pp. 227–229.

Monge, G. (1781). Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*.

Parikh, N. and Boyd, S. (2014). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), pp. 127–239.

Peyré, G. and Cuturi, M. (2019). Computational Optimal Transport. *Foundations and Trends in Machine Learning*, 11(5–6), pp. 355–607.

Vayer, T., Chapel, L., Flamary, R., Tavenard, R., and Courty, N. (2019). Optimal Transport for Structured Data with Application on Graphs. *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Villani, C. (2008). *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften, vol. 338. Springer.