Optimizing last-mile delivery through crowdshipping on public transportation networks

Bianchi Ludovica

University of Trieste

February 2, 2026

Contents

Outline

Problem Motivation

The rapid growth of e-commerce and on-demand services is intensifying **last-mile delivery challenges** in urban areas:

- increased traffic congestion, pollution, and noise,
- higher operational costs for logistics providers,
- pressure on urban infrastructure.

Emerging solutions include:

- adoption of new delivery technologies (e.g., drones, autonomous robots),
- integration of freight transportation with existing public transport systems.

A new delivery paradigm

Core idea: involving regular commuters in the delivery process.

Context: urban and suburban networks with dense deployment of Automated Parcel Lockers (APLs) and high commuter flows.

How does it work?

- parcels travel through the public transport network,
- commuters transport parcels along their regular trips,
- deliveries are completed at peripheral lockers,
- reducing last-mile traffic in city centers.

PTCP PROBLEM

Optimisation's goal: optimize parcel delivery from the LSP warehouse to destination lockers using public transport and commuters.

Decisions include:

- selecting *source stations* served by LSP trucks,
- selecting feasible *destination stations* among those requested by customers,
- assigning parcels to *crowdshippers* without route deviations,
- satisfying APL capacity constraints,
- using a backup delivery service when necessary.

Advantages and Model Innovation

Advantages:

- **Economic**
  - commuters receive travel incentives,
  - logistics providers reduce operational costs.
- **Environmental**
  - reduced last-mile vehicle traffic,
  - lower emissions in urban areas.

Model innovation: integration in a unique model of different features:

1. multi-step parcel routing via multiple commuters,
2. joint evaluation of crowdshipping and backup delivery costs.

This work presents:

1. a mixed-integer linear programming (MILP) formulation for the PTCP problem,
2. two strengthened MILP models using valid inequalities:
   - dynamically separated via callbacks,
   - statically integrated into the formulation,
3. an Adaptive Large Neighborhood Search (ALNS) metaheuristic.

Outline

Problem definition and notation

1. PT NETWORK

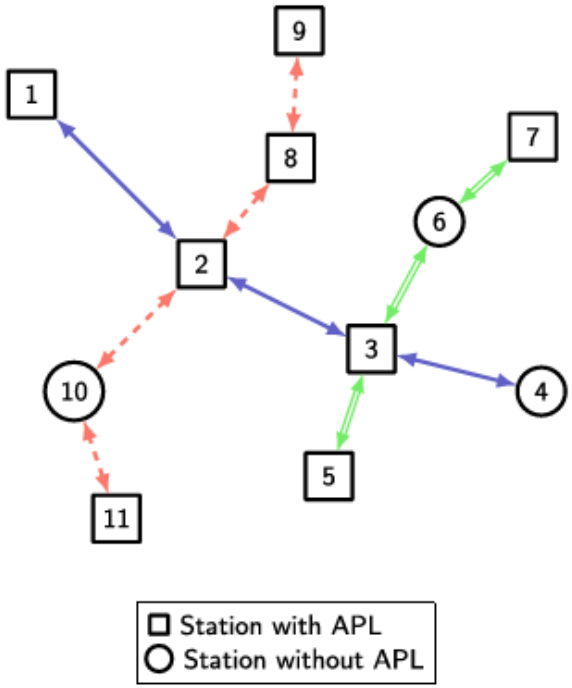
- $L = \{1, ..., l_{max}\}$: set of transportation lines;
- $\bar{N}$: set of network stations equipped with APLs;
- $\bar{S}$: set of source stations;
- $\bar{D}$: set of destination stations s.t. $\bar{S} \cap \bar{D} = \emptyset$;
- Exchange stations: $i \in \bar{N}$ s.t. $\exists \mathcal{L}_i \subseteq L : |\mathcal{L}_i| \geq 2, \cap_{i \in \mathcal{L}_i} \bar{N}_l \neq \emptyset$

TRANSFER GRAPH: $\bar{G} = (\bar{N}, \bar{A})$ where:

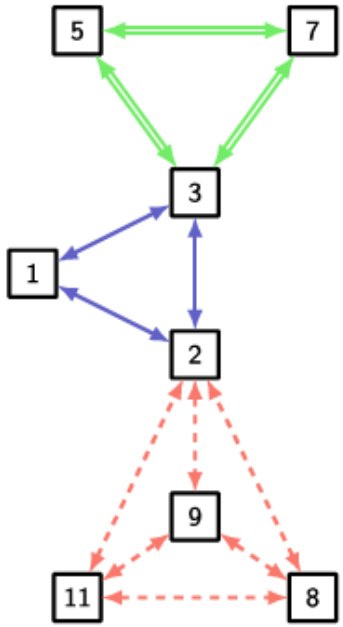
$$\bar{A}_l = \{(i,j) \mid i,j \in \bar{N}_l\}, \qquad \bar{A} = \bigcup_{l \in L} \bar{A}_l.$$

Example

Example of a small PT network consisting of three transportation lines and 11 stations, where the nodes drawn as squares represent network stations equipped with APLs.



(a) Real network



(b) Transfer graph $\bar{G} = (\bar{N}, \bar{A})$

2. CROWDSHIPPERS

K: set of crowdshippers such that:

- each crowdshipper follows the shortest path between their entry and exit stations without deviating from their planned route;
- $N_k \subset \bar{N}$: stations visited by crowdshipper k where parcels can be handled. In particular:
  - parcels can be picked up at entry or exchange stations;
  - parcels picked up at an exchange station can be delivered either at the exit station or at another exchange station.
- $K_{ij} \subseteq K$: subset of crowdshippers who can transfer a parcel from station i to station j.

CROWDSHIPPER DELIVERY GRAPH: $G_k = (N_k, A_k)$ where:

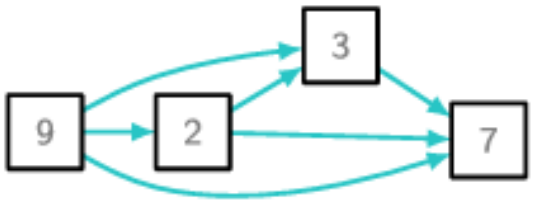
$$A_k = \{(i,j) \in \bar{A} \mid i,j \in N_k\}$$

represents all feasible parcel transfers for crowdshipper k.

Example

Routes for three crowdshippers:

1. CS 1: origin= 9, dest= 7, shortest path= $\{9, 2, 3, 7\}$
2. CS 2: origin= 1, dest= 7, shortest path= $\{1, 3, 7\}$
3. CS 3: origin= 11, dest= 7, shortest path= $\{11, 2, 7\}$



(a) Delivery graph G_1 for Crowdshipper 1

(b) Delivery graph G_2 for Crowdshipper 2

(c) Delivery graph G_3 for Crowdshipper 3

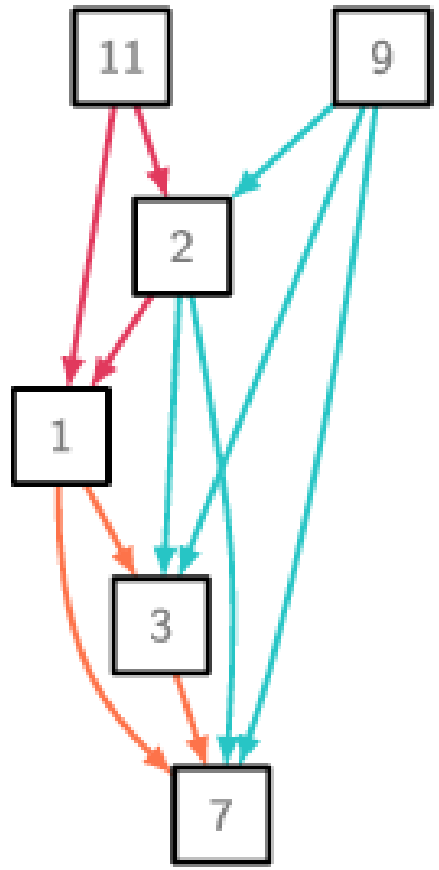
3. PARCELS

- P: set of parcels;
- $D_p \subseteq \bar{D}$: feasible destination stations selected by the recipient of parcel p.

GLOBAL DELIVERY GRAPH:
$G = (N, A)$ where:

$$N = \bigcup_{k \in K} N_k, \qquad A = \bigcup_{k \in K} A_k,$$

represent all stations and transfers reachable through crowdshippers.



(d) Graph $G = (N, A)$

TIME, COSTS AND PARAMETERS:

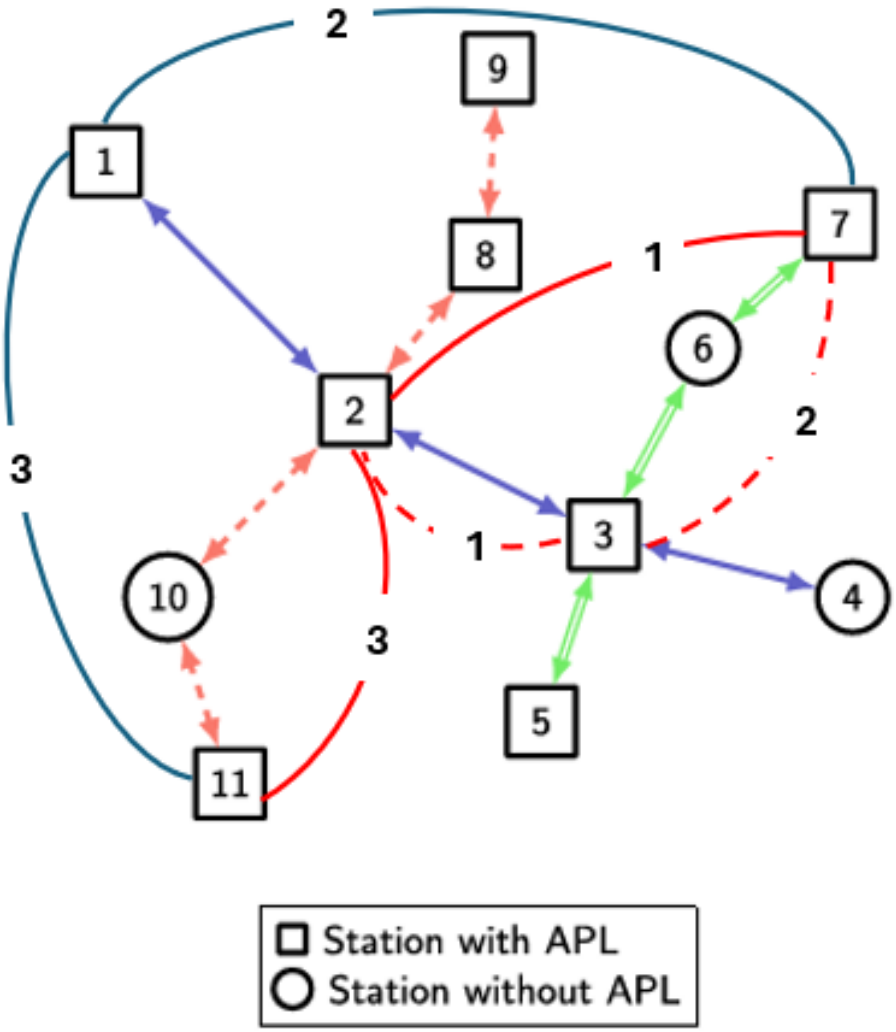
4. TIME DISCRETIZATION:

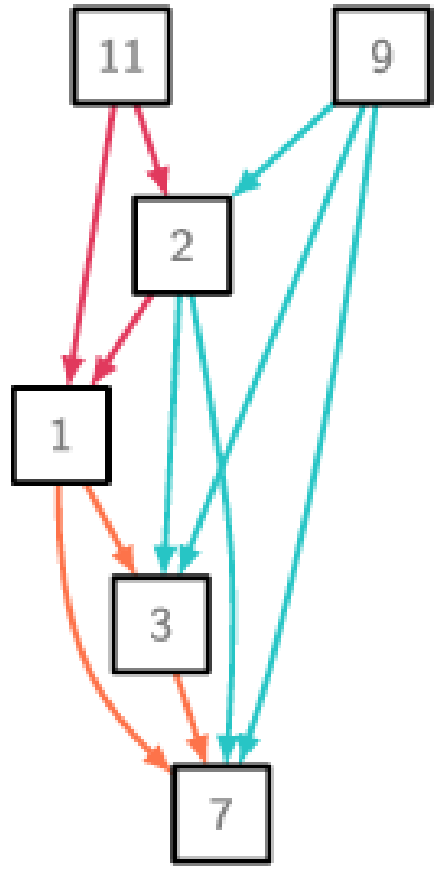
- T: set of time slots into which the delivery dai is divided;
- t_{ij}:cumulative travel time between i and j in the transfer graph.
- K_i^t: set of crowdshippers present at network station i at time slot t.

5. COSTS:

- ρ: reward paid by the LSP to crowdshippers for performing a delivery;
- ν_i: fixed cost of sending an LSP vehicle to source station i;
- σ_i: unit cost for loading a parcel into a source staton i;
- γ_p: backup delivery cost for parcel p;
- C_i: capacity of the APL located at station i.

Example





(d) Graph $G = (N, A)$

Decision variables

$$x_{ij}^{kp} = \begin{cases} 1 & \text{if crowdshipper } k \text{ transfers parcel } p \text{ from } i \text{ to } j, \\ 0 & \text{otherwise} \end{cases}$$

$$z_k = \begin{cases} 1 & \text{if crowdshipper } k \text{ takes a parcel,} \\ 0 & \text{otherwise} \end{cases}$$

$$o_{ipt} = \begin{cases} 1 & \text{if parcel } p \text{ is at station } i \text{ at time slot } t, \\ 0 & \text{otherwise} \end{cases}$$

$$a_{ipt} = \begin{cases} 1 & \text{if parcel } p \text{ arrives at station } i \text{ at time slot } t, \\ 0 & \text{otherwise} \end{cases}$$

$$l_{ipt} = \begin{cases} 1 & \text{if parcel } p \text{ leaves station } i \text{ at time slot } t, \\ 0 & \text{otherwise} \end{cases}$$

$$y_{kp} = \begin{cases} 1 & \text{if parcel } p \text{ is assigned to crowdshipper } k, \\ 0 & \text{otherwise} \end{cases}$$

$$s_{ip} = \begin{cases} 1 & \text{if parcel } p \text{ starts its route at source station } i, \\ 0 & \text{otherwise} \end{cases}$$

$$d_{ip} = \begin{cases} 1 & \text{if parcel } p \text{ ends its route at destination station } i, \\ 0 & \text{otherwise} \end{cases}$$

$$e_i = \begin{cases} 1 & \text{if source station } i \text{ is selected}, \\ 0 & \text{otherwise} \end{cases}$$

$$q_p = \begin{cases} 1 & \text{if parcel } p \text{ is delivered by the backup service}, \\ 0 & \text{otherwise} \end{cases}$$

Objective function

Minimize Total Operational Cost

$$\min \rho \sum_{k \in K} z_k + \sum_{i \in S} \nu_i e_i + \sum_{i \in S} \sigma_i \sum_{p \in P} s_{ip} + \sum_{p \in P} \gamma_p q_p \quad (1)$$

Where components are: (1) Crowdshipper activation, (2) Station fixed cost, (3) Loading cost, (4) Backup cost.

Constraints

Unique source and destination constraint

$$\sum_{i \in S} s_{ip} = 1 - q_p \qquad p \in P \qquad (2)$$

$$\sum_{i \in D_p} d_{ip} = 1 - q_p \qquad p \in P \qquad (3)$$

Activation station and flow constraints

$$\sum_{p \in P} s_{ip} \leq C_i e_i \qquad i \in S \qquad (4)$$

$$\sum_{j \in N} \sum_{k \in K_{ij}} x_{ij}^{kp} - \sum_{j \in N} \sum_{k \in K_{ji}} x_{ji}^{kp} = s_{ip} - d_{ip} \qquad p \in P, i \in N \qquad (5)$$

Crowdshipper activation constraint

$$x_{ij}^{kp} \leq y_{kp} \qquad p \in P, k \in K, (i,j) \in A_k \qquad (6)$$

Each crowdshipper one parcel

$$\sum_{p \in P} y_{kp} \leq z_k \qquad k \in K \qquad (7)$$

Temporal coherence constraints

$$l_{ipt} = \sum_{j \in N} \sum_{k \in K_i^t} x_{ij}^{kp} \qquad p \in P, i \in N, t \in T \qquad (8)$$

$$a_{ipt} = \sum_{j \in N} \sum_{k \in K_j^t} x_{ji}^{kp} \qquad p \in P, i \in N, t \in T \qquad (9)$$

$$o_{ip(t+1)} \geq a_{ipt} \qquad i \in N, p \in P, t \in \{1, \ldots, |T| - 1\} \qquad (10)$$

$$o_{ip(t+1)} \geq o_{ipt} - l_{ipt} \qquad i \in N, p \in P, t \in \{1, \ldots, |T| - 1\} \qquad (11)$$

$$o_{ip(t+1)} \leq 1 - l_{ipt} \qquad i \in N, p \in P, t \in \{1, \ldots, |T| - 1\} \qquad (12)$$

One location per slot constraint

$$\sum_{i \in N} o_{ipt} \leq 1 - q_p \qquad p \in P, t \in T \qquad (13)$$

Capacity constraint

$$\sum_{p \in P} o_{ipt} + \sum_{p \in P} a_{ipt} \leq C_j \qquad i \in N, t \in T \qquad (14)$$

Initial constraint

$$o_{ip1} = s_{ip} \qquad p \in P, i \in N \qquad (15)$$

Valid Inequalities (1/2)

Introduced to strengthen the relationships among the variables,to improve the model's structure and optimization performance.

$$\sum_{p \in P} s_{ip} \geq e_i \qquad i \in N \qquad (24)$$

$$\sum_{i \in D} d_{ip} \leq \sum_{k \in K} y_{kp} \qquad p \in P \qquad (25)$$

$$\sum_{i \in S} s_{ip} \leq \sum_{k \in K} y_{kp} \qquad p \in P \qquad (26)$$

$$\sum_{i \in N} \sum_{t \in T} l_{ipt} \leq \sum_{k \in K} y_{kp} \qquad p \in P \qquad (27)$$

$$\sum_{i \in N} \sum_{t \in T} a_{ipt} \leq \sum_{k \in K} y_{kp} \qquad p \in P \qquad (28)$$

Valid Inequalities (2/2)

$$y_{kp} \leq 1 - q_p \qquad p \in P, k \in K \qquad (29)$$

$$s_{ip} + d_{ip} \leq 1 \qquad i \in N, p \in P \qquad (30)$$

$$\sum_{i \in N} o_{ipt} \leq \sum_{i \in S} s_{ip} \qquad p \in P, t \in T \qquad (31)$$

$$\sum_{i \in N} o_{ipt} \leq \sum_{i \in D} d_{ip} \qquad p \in P, t \in T \qquad (32)$$

Outline

Large Neighborhood Search (LNS) (Shaw (1998))

- Iterative improvement heuristic based on **destroy-and-repair** moves.
- At each iteration, a subset of requests (q) is removed from the current solution s.
- The removed requests are then reinserted, possibly in a different way, to obtain a new solution.
- Enables exploration of a very large neighborhood of solutions' space in limited computational time.

Main limitations (from Ropke & Pisinger (2006)):

- The solution is modified only locally at each iteration.
- Difficulties in escaping local optima in tightly constrained problems.
- Strong dependence on the choice of destroy and repair operators.
- The parameter q controls the trade-off between intensification and diversification.

How can we make LNS more robust and adaptive?

Adaptive Large Neighborhood Search (ALNS)

Extension by Ropke & Pisinger (2006):

- Extends LNS by allowing multiple destroy/repair operators to compete.
- **Adaptive Layer:** operators are selected dynamically based on their past success rate.

Operator Selection Mechanism

1. Reference Paper (Gajda et al., 2025):

- Standard **roulette wheel selection** based solely on weights w_i.

2. Our Implementation (ϵ-greedy strategy): introduce a stochastic component to balance *exploitation* and *exploration*:

- **Exploitation ($1 - \epsilon = 90\%$):** select operator j based on performance weights (Roulette-wheel):

$$\mathbb{P}(\text{select op } j) = \frac{w_j}{\sum_i w_i}$$

- **Exploration ($\epsilon = 10\%$):** select an operator **uniformly at random**.
  - *Rationale:* prevents the algorithm from stagnating in a subset of operators and ensures all neighborhoods are occasionally visited.

Adaptive Learning: scoring & weight update

Scoring system: Operators receive a score σ based on the move outcome (Ropke & Pisinger (2006)):

- σ_1 (high): new global best solution found.
- σ_2 (medium): improvement over current solution.
- σ_3 (low): accepted non-improving solution.

Weight Update Mechanism Comparison:

Reference Paper

- **Segmented Update:** search is divided into fixed segments.
- **Hard reset:** weights are reset after each segment.
- Focus more on short-term performance.

Our Implementation

- **Continuous Update (EMA):**

$$w_{j,t+1} = (1 - \eta) \cdot w_{j,t} + \eta \cdot \frac{\text{Score}_j}{\text{Usage}_j}$$

where η is the reaction factor.

- avoids abrupt resets and preserves long-term knowledge of operator performance.

Acceptance criterion & temperature management

Algorithm structure: ISA vs. SA

- Paper $\longrightarrow$ **ISA (Iterated SA):** double-loop structure with resets every each fixed number of iterations.
- Ours $\longrightarrow$ **Continuous SA:** single loop with stagnation detection to escape local minima.

Acceptance rule: a candidate solution S' is accepted with probability:

$$\exp\left(-\frac{\Delta}{T}\right)$$

Adaptive initialization (Talbi (2009)/Aarts & Korst (1988))

- Paper $\longrightarrow$ Fixed/Generic parameter (not specified).
- Ours $\longrightarrow$ **Instance-Calibrated:**
  - Sample *n samples* random moves to estimate landscape "roughness" (Δ_{avg}).
  - T_0 set to accept a worsening of Δ_{avg} with prob. 80%.

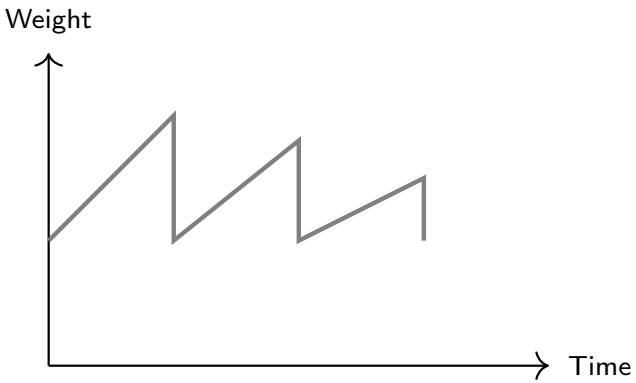
2. Cooling & proactive reheating

- Standard geometric cooling: $T \leftarrow \alpha \cdot T$.
- **Reheating (stagnation-based):** triggered after N non-improving iters.
  1. Reset search to *Best Known Solution*.
  2. Boost temperature ($T \uparrow$) to enable escape.
  3. apply a strong diversification (kick) move.

Why EMA? Weight Evolution Comparison

Conceptual comparison of operator knowledge retention

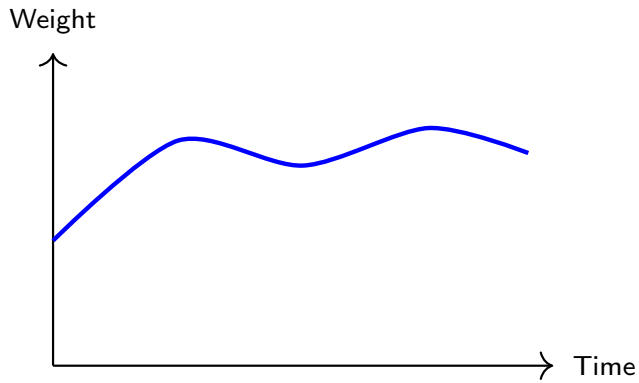
Segmented (Paper)



"Sawtooth": Knowledge lost at reset

Weights oscillate and reset.
Short-term memory only.

Continuous EMA (Ours)



Smooth adaptation

Weights evolve smoothly.
Balances history and recent trends.

Conclusion EMA provides a more stable learning process, preventing the algorithm from discarding valid operators simply because of a bad short-term segment.

Destroy heuristics

Objective:

- Remove a subset of parcel routes from the current solution.
- Move them to a backup set to enable diversification.

Station load representation

$$H \in \mathbb{N}^{|N| \times |T|}, \quad h_{it} = \text{parcels at station } i \text{ at time } t$$

Station prioritization criteria: sort stations according to

1. Absolute load: $a_i = \max_t h_{it}$,
2. Relative load: $r_i = a_i / C_i$,
3. Average load: $m_i = \frac{1}{|T|} \sum_t h_{it}$,
4. Random ordering (diversification).

How can we destroy the selected paths?

(D1) Parcel Path Elimination

1. Select the highest-priority stations.
2. Sort their parcel paths by non-increasing length.
3. Remove up to $q^* = \min(\bar{q}, q)$ longest paths and move them to backup (q is the pre-specified number of paths to eliminate).

(D2) Capacity Reduction

1. Select the top q stations.
2. Reduce capacity: $\tilde{C}_i = \max(C_i - \tilde{q}, 0)$.
3. Iteratively remove longest paths until feasibility is restored for each time step.

Repair heuristics

Objective:

- Deliver backup parcels using crowdshipping.
- Construct feasible delivery paths.

Method: Label-setting algorithm

Each station j is associated with a label (c_j, t_j, k, i) where:

- c_j: cumulative cost to reach station j,
- t_j: arrival time at station j of crowdshipper k from station i,
- k: crowdshipper under consideration,
- i: predecessor station in the path.

Label-setting algorithm

Initialization: $(0, 0, \emptyset, \emptyset)$

Step: from label $(c_i, t_i, \cdot, \cdot)$, for each neighbor j and feasible crowdshipper k:

- $c_j = c_i + f_k(i, j)$,
- $t_j = t_i + t_{ij}^k$.

Dominance rule: a new label replaces the existing one if it has lower cost. Two alternative cost functions are considered:

1. Uniform cost: $f_k(i, j) = 1$
   $\rightarrow$ minimizes the number of employed crowdshippers.
2. Load-aware cost: arc cost equals the historical usage of the crowdshipper $\rightarrow$ promotes load balancing.

Termination: when a label at the destination station is fixed.

In which order are parcels processed?

Backup parcels are processed according to:

1. As-is order,
2. Random order,
3. Descending backup cost (Max Backup),
4. Ascending backup cost (Min Backup).

Repair heuristics structure

Each repair heuristic consists of two phases:

1. **Source station assignment**
   - based on marginal loading cost $p_i(n)$, or
   - based on absolute loading cost $p_i(0)$ and on the nearest feasible destination heuristic.
2. **Path construction**
   - minimize number of crowdshippers, or
   - balance crowdshipper usage.

Repair operators

- **R1**: Parcel distribution + shortest path
- **R2**: Delivery cost minimization + shortest path
- **R3**: Parcel distribution + load balancing
- **R4**: Delivery cost minimization + load balancing

Constructive Initial Heuristic (CIH)

- Starts from a full backup solution.
- Iteratively repairs parcels using a greedy strategy.
- Uses repair heuristic R1.
- Parcels are processed in As-Is order.
- Uniform arc costs are used to minimize the number of crowdshippers.

Outline

Tests and Scalability — MILP benchmark

- **Objective:** evaluate scalability and effectiveness of three MILP variants (Base, Callback, Cuts) on 9 families of instances;
- **Focus:** identify where MILP is usable, analyze cuts/callback behavior, and establish a baseline for ALNS comparison.
- What we measure:
  - **Runtime** (s);
  - **Time-to-best** (TTB);
  - **Optimality gap** and **Objective value**;
  - **Root bound after cuts:** demonstrates bound strengthening at the root node;
  - **Nodes / Memory:** indicators of practical scalability;
  - **Cuts statistics:** counts by family groups (for callback variant).

Experimental setup

1. **Instances:** 3 network sizes ($|N| = 24, 36, 44$) $\times$ 5 load classes ($|K|, |P|$) aggregated $\rightarrow$ 15 families (45 instances total);
2. **Solvers:**
   - *MILP Base*;
   - *MILP Callback* (dynamic cuts);
   - *MILP Cuts* (static pre-generated cuts);
3. **Time limit:** 3600 s, (OOM/timeouts considered as 3600s, gap = 1.0);
4. **Reporting:** Aggregated means per tuple ($|N|, |P|, |K|$).

MILP Baseline

2. Solution Validity (Sanity Check):

Class	$\|P\|$	$\|K\|$	Time (s)	Gap (%)	% Delivered
Small	30	50	28.4	**0.0**	**78.9%**
Med	$\geq 50, \leq 75$	$\geq 75, \leq 100$	173.4	**0.0**	70.1%
Large	≥ 120	≥ 150	*3600 (TL)*	> 30.0	52.6%

Table: aggregated performance of MILP Base

- Small instances ($P = 30, K = 50$): solved fast with gap 0;
- Medium instances: runtime increases but still gap 0;
- Large instances ($P = 120, K = 150$ and $\geq 160/ \geq 200$): dramatic runtime increase and non-zero gaps (timeouts).

MILP Callback & MILP Cuts

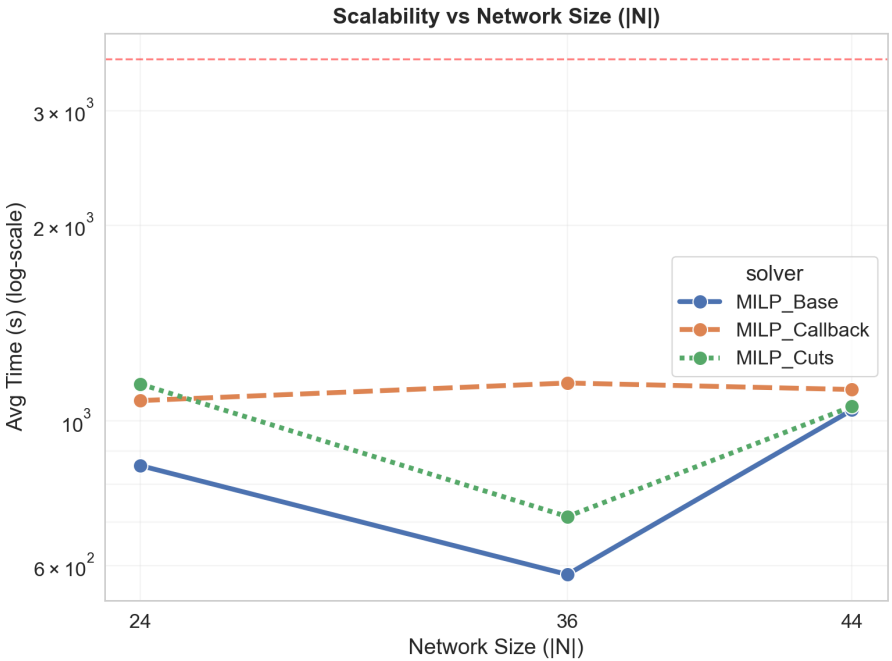
Table: Impact of Load ($|P|$) on Solver Metrics (fixed $|N| = 36$)

Regime ($\|P\|, \|K\|$)	**Presolve Time (s)** *Sim*	*Cbk*	**Cut**	**Nodes Explored** *Sim*	**Cbk**	*Cut*	**Root Gap** *All*
Low Load (30, 50)	1.9s	1.6s	3.4s	~ 2	3	3	$\approx 0\%$
	No significant overhead			*Trivial search*			*Solved*
Med Load (75, 100)	8.5s	21.2s	12.3s	280	**37**	558	$< 2\%$
	Cuts add overhead			**Cbk cuts tree by 87%**			*Solved*
High Load (120, 150)	44s	18s	**76s**	692	814	1270	**18-34%**
	Static Cuts heavy init			*Struggling to branch*			*Unstable*

- In medium instances the Cut variant shows lower optimality gaps.
- While dynamic cuts (Cbk) drastically reduce the search tree size, the cut generation overhead often outweighs the branching savings, resulting in higher wall-clock times compared to the Baseline

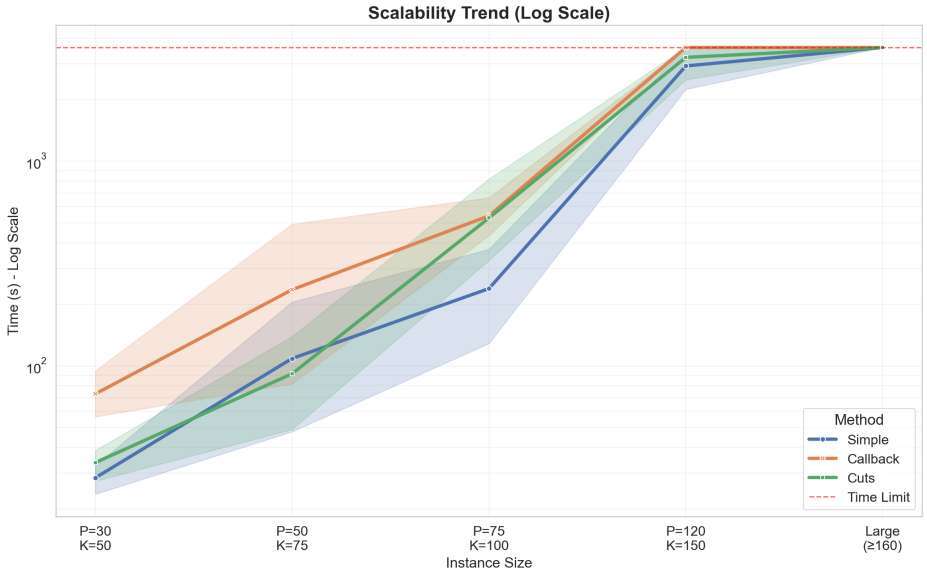
Scalability Analysis: Network vs. Density

A. Network Size ($|N|$)



Runtime is stable across map sizes.
Topology is **not** the bottleneck.

B. Density Load ($|P|, |K|$)



Exponential explosion on Log Scale.
Combinatorial matching saturates the solver.

Cuts analysis: how many cuts are generated?

TABLE CUTS VI ANALYSIS

|N|	|P|	|K|	VI1	VI2-VI3	VI4-VI6	VI7-VI9	Total
24	30	50	0.0	31.3	43.3	0.0	74.7
36	30	50	0.0	36.0	45.3	0.0	81.3
44	30	50	0.0	36.7	40.0	0.0	76.7
24	50	75	0.0	64.0	82.0	0.0	146.0
36	50	75	0.0	89.3	131.7	0.0	221.0
44	50	75	0.0	130.7	157.7	0.0	288.3
24	75	100	0.0	99.3	160.7	0.0	260.0
36	75	100	0.0	207.3	273.3	0.0	480.7
44	75	100	0.0	184.0	221.0	0.0	405.0
24	120	150	0.0	220.0	298.7	0.0	518.7
36	120	150	0.0	80.0	206.0	0.0	286.0
44	120	150	0.3	264.7	333.7	0.0	598.7
24	>=160	>=200	0.0	37.0	51.5	0.0	88.5
>=36	>=160	>=200	0.0	158.4	179.6	0.0	338.0

Why are certain cuts not considered?

Analysis of Inactive Cuts (Redundant)

- **(VI1)** $\sum_{p \in P} s_{ip} \geq e_i \quad i \in N$:
  - Dominated by optimality costs (solver naturally minimizes e_i to 0 if no parcels).
  - Logically implied by capacity constraint (4).
- **(VI7)** $s_{ip} + d_{ip} \leq 1 \quad i \in N, p \in P$:
  - Trivial: disjointness of sources and destinations by construction.
- **(VI8-VI9)** $\sum_{i \in N} o_{ipt} \leq \sum_{i \in S} s_{ip}, \quad \sum_{i \in N} o_{ipt} \leq \sum_{i \in D} d_{ip}$
  - Logically implied by unique source/destination constraints (2) and (3), as total inventory cannot exceed 1.

Observation These cuts do not significantly tighten the LP polyhedron and can be disabled to reduce generation overhead.

Analysis of Active Cuts (Effective)

1. Global Cardinality Constraints (VI4-VI5):

$$\sum_{i \in N} \sum_{t \in T} l_{ipt} \leq \sum_{k \in K} y_{kp} \quad \text{and} \quad \sum_{i \in N} \sum_{t \in T} a_{ipt} \leq \sum_{k \in K} y_{kp}$$

Rationale:

- **Aggregation:** Reduces $O(|P| \cdot |N| \cdot |T|)$ local constraints to $O(|P|)$ global bounds.
- **Linking:** Bridges time-dependent flow (l, a) and strategic assignment (y).
- **Cut Effect:** Cuts LP solutions where parcels perform "too many hops" relative to assigned crowdshippers.

2. Connectivity Cuts (VI2-VI3):

$$\sum_{i \in S} s_{ip} \leq \sum_{k \in K} y_{kp} \quad \text{and} \quad \sum_{i \in D} d_{ip} \leq \sum_{k \in K} y_{kp}$$

- These prevent the LP relaxation from "satisfying" demand at the source (avoiding backup costs) without paying the activation cost of at least one crowdshipper.

Final Takeaways

- **Small families:** exact MILP solves them reliably (gap 0%, moderate time).
- **Medium instances:** results vary. MILP Cuts sometimes speeds up the solution, while MILP Callback often costs more CPU time without converging faster.
- **Large-load families** ($P \geq 120, K \geq 150$): all MILP variants struggle (timeouts and high gaps).
- **Cuts insight:** a small subset of VI families is dominant (VI2–VI3, VI4–VI6). Unused families offer an opportunity to simplify callbacks logic.
- **Conclusion:** MILP Base is the most robust baseline for reproducible comparisons against ALNS.

Outline

Aggregate performance overview

Here we focus on scalability and convergence behavior:

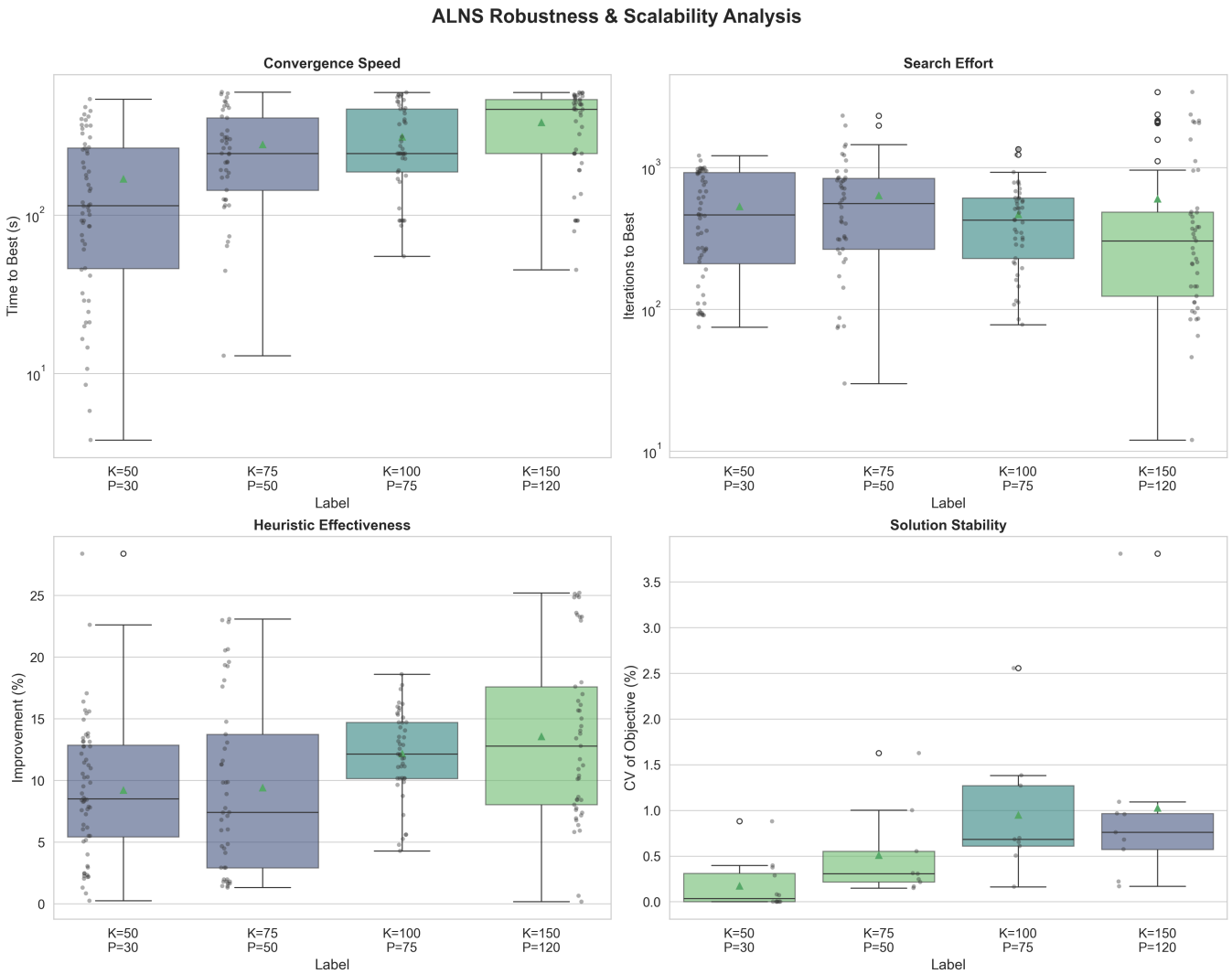
- Multiple random seeds per instance;
- Comparison based on averaged performance across the runs and then average over instances belonging to the same class.

Size	$\|P\|$	$\|K\|$	Obj (Avg)	% CS Del.	TTB [s]	Impr. %
Small	30	50	124.7	79.6%	181	8.7%
Med	50	75	256.9	63.7%	278	5.1%
Large	75	100	331.4	76.2%	312	12.1%
X-Large	120	150	476.1	76.8%	386	13.5%

Key Takeaways:

- **Feasibility:** feasible solutions found for all instances.
- **Crowdshipping usage:** consistently high, proving the algorithm effectively exploits the crowd network.
- **Scalability:** TTB grows smoothly with instance size. Even largest instances solved in < 7 minutes.

Robustness & Search Behavior



Convergence:

- **Time to Best:** increases moderately with problem size, expected behavior.
- **Improvement:** positive and consistent 5-15% gain over constructive heuristic.

Stability:

- **Search Effort:** remains comparable across instance sizes.
- **Robustness:** very low CV of objective ($< 1.5\%$) across seeds, suggesting stable performance across runs.

Outline

MILP vs ALNS: what we measure

Goal: Validate the heuristic against the exact solver on solvable instances.

- **Solution Quality:**
  - *Optimality Gap:* Deviation from MILP optimal solution (Best/Mean/Worst across seeds).
  - *Match Rate:* % of ALNS runs finding the exact optimal solution.
- **Efficiency:**
  - *Time-to-Best (TTB):* Wall-clock time to reach the best found solution.
- **Dynamics:**
  - *Convergence History:* Evolution of the objective value over time to understand search behavior.

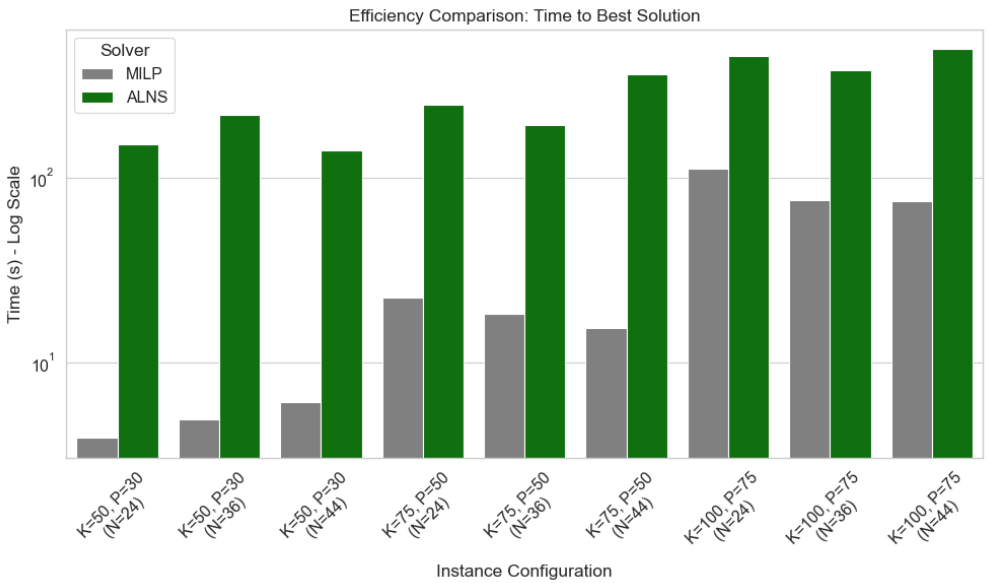
Solution Quality Analysis

Table: Gap Analysis: ALNS deviation from MILP optimum

|K|	|P|	|N|	Match Rate	Gap_Best	Gap_Mean	Gap_Worst
50.0	30.0	24.0	25.0	4.51	4.65	5.04
50.0	30.0	36.0	25.0	0.99	1.14	1.36
50.0	30.0	44.0	55.0	0.37	0.48	0.63
75.0	50.0	24.0	10.0	0.13	3.48	5.32
75.0	50.0	36.0	15.0	0.17	2.1	4.24
75.0	50.0	44.0	20.0	1.54	2.58	3.34
100.0	75.0	24.0	6.67	0.45	3.77	5.77
100.0	75.0	36.0	13.33	0.12	3.93	7.55
100.0	75.0	44.0	6.67	0.91	4.04	7.03

- **High Accuracy:** The mean Gap is consistently low ($\approx 1\% - 4\%$) even for the largest tested instances ($N = 44, P = 75$).
- **Match Rate:** on small instances ($N = 24$), ALNS finds the exact optimum frequently (up to 55% match rate).
- **Reliability:** The gap variability (mean vs worst) remains contained.

Efficiency Comparison: Time-to-Best



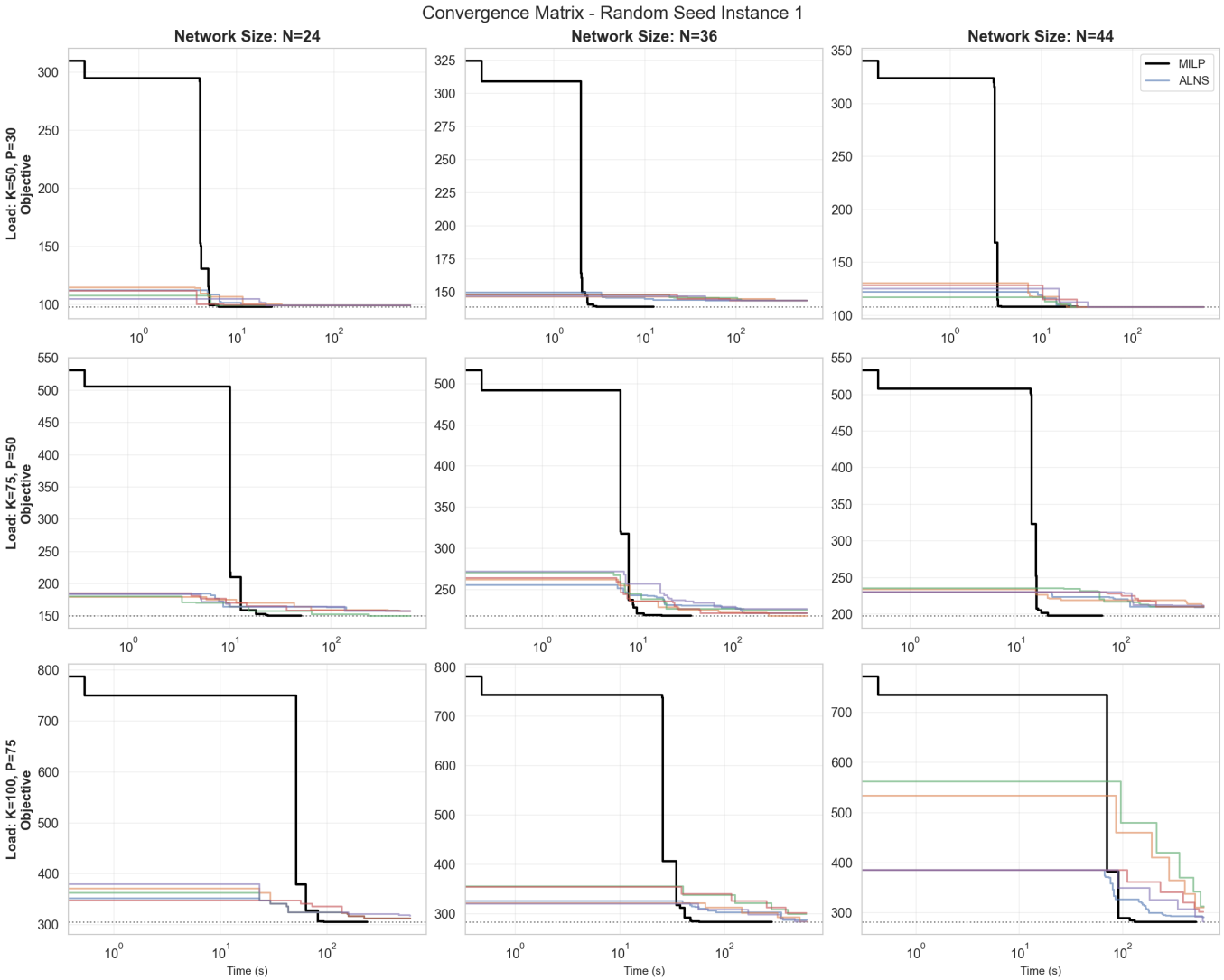
MILP dominance on small sets:

- For solvable instances ($N \leq 44$), Gurobi is significantly faster than ALNS.
- *Reason:* on limited search spaces, exact pruning is more efficient than iterative sampling.

ALNS Role:

- ALNS has a "warm-up" cost (initial iterations).
- **Value:** while slower on small maps, ALNS is proposed as a viable option when MILP times out.

Convergence Dynamics: stepwise vs. continuous



Outline

Conclusions & Key Findings

1. **To sum up:**

   - Integration of public transport into parcel delivery via crowdshipping (commuters + lockers), supported by a backup service.
   - Formulated as a **MILP** with strict synchronization constraints.
   - Developed two solvers: **Branch-and-Cut** (Exact) and **ALNS** (Metaheuristic).

2. **Algorithmic Performance**

   - **Exact Method:** efficient only on small instances; struggles with scalability due to combinatorial explosion.
   - **ALNS Efficiency:** consistently yields high-quality solutions (Gap $\leq 5\%$) in < 7 **minutes**, suggesting to test for realistic large-scale scenarios.

To investigate:

1. **Sensitivity analysis insights**
   - **Cost ratio impact:** The balance between crowdshipping remuneration and backup costs is the primary driver of system profitability.
   - **Network Capacity:** locker availability significantly constrains the flow; saturation requires efficient routing to avoid backup overuse.
2. **Advanced Pricing & Operational Flexibility**
   - **Dynamic Incentives:** moving from fixed payments to *distance-based rewards* or *dynamic travel credits* linked to demand/capacity.
   - **Limited Diversions:** allowing commuters to deviate slightly from their path to increase match rates.

Future directions: towards real-world applicability

1. From Deterministic to Stochastic (delay analysis)

- Current model assumes perfect PT adherence. Future work must address:
  - **Public transport delays:** stochastic travel times to model missed connections.
  - **Robust optimization:** buffer times and dynamic re-routing to handle disruptions.
  - **Demand uncertainty:** stochastic parcel arrival rates vs. crowdshipper availability.

2. Technological integration

- Real-time synchronization with **Live PT Schedules** (GTFS-Realtime).
- Co-design of dedicated infrastructure (Lockers) with Public Transit Authorities.

Outline

Heuristic Tuning: Setup & Effectiveness

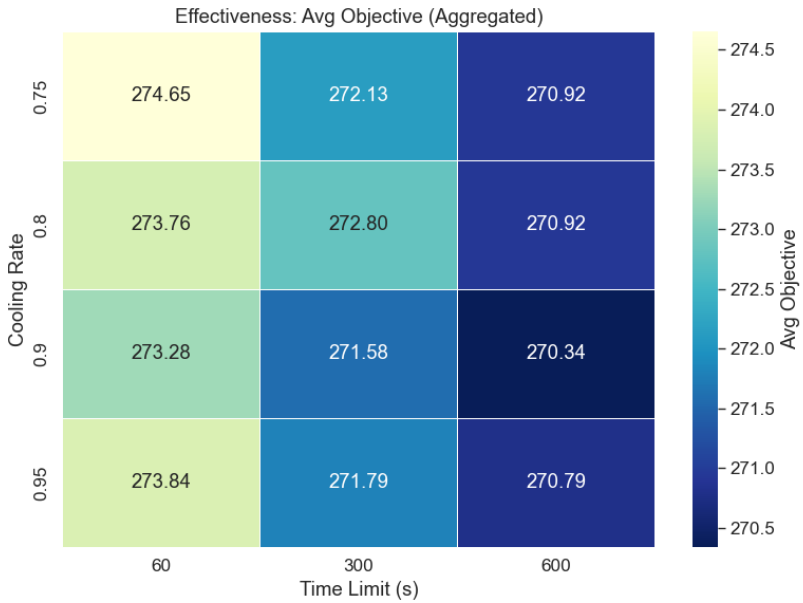
Goal: Balance quality vs. efficiency.

Grid Search:

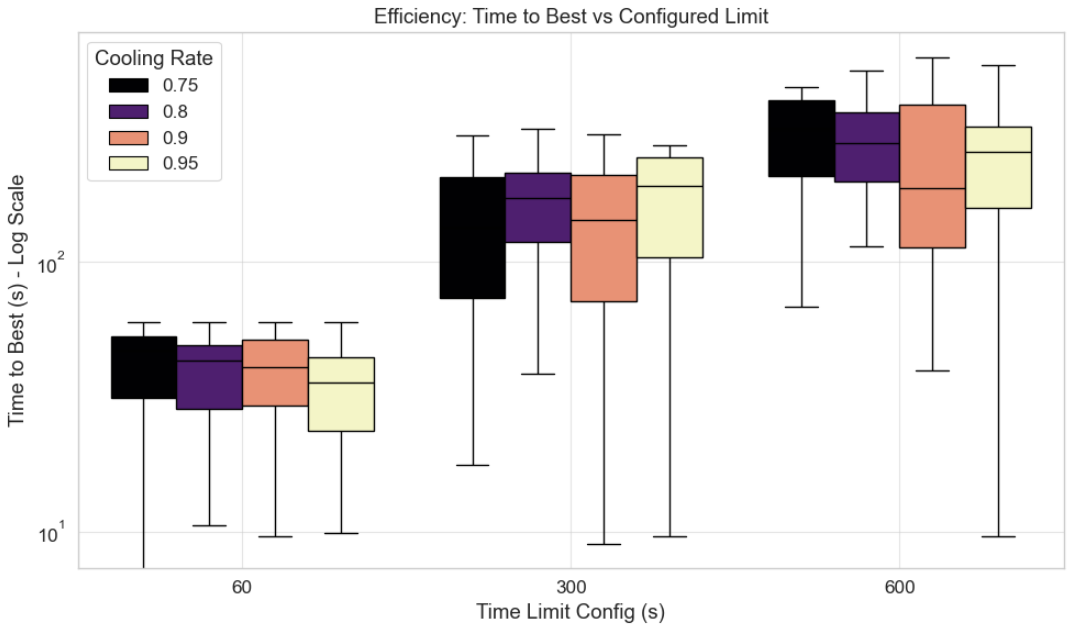
- **Cooling (α):** $\{0.75, 0.80, 0.90, 0.95\}$
- **Time (T_{max}):** $\{60s, 300s, 600s\}$

Best Config Found:
$\alpha = 0.90$, $T = 600s$

Avg. Objective (Lower is better)

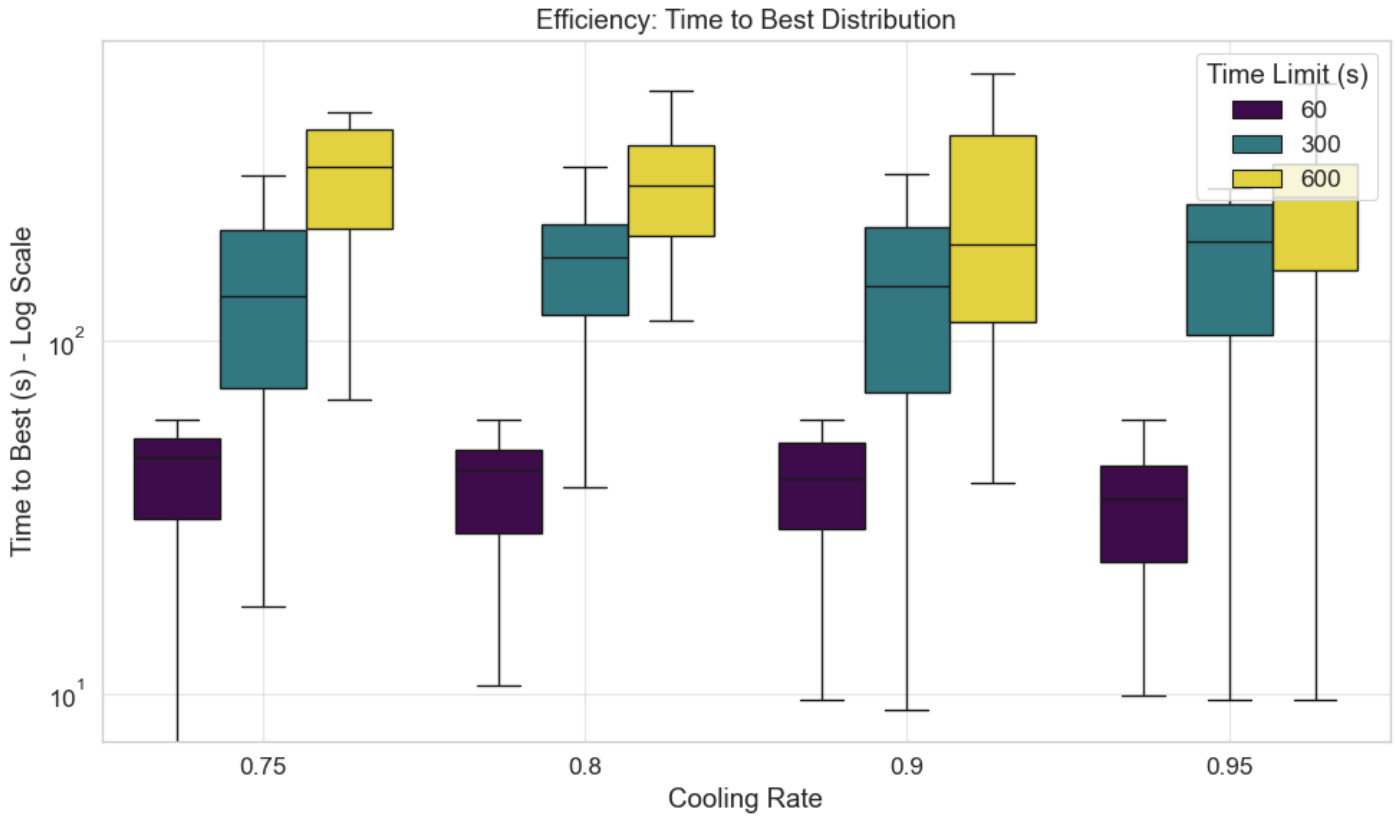


Efficiency Analysis: Time to Best (TTB)



- **High Variance at 600s:** short runs often get stuck in local optima (high spread).
- **Trade-off:** We accept a higher TTB to gain solution quality and reliability.

Efficiency Analysis: Time to Best (TTB)



Essential References


Mikele Gajda, Olivier Gallay, Renata Mansini, Filippo Ranza (2025) *Optimizing last-mile delivery through crowdshipping on public transportation networks*, Transportation Research Part C: Emerging Technologies, **179**.


Stefan Ropke, David Pisinger (2006) *An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows*, Transportation Science, **40**.