Pickup and delivery problem with drones and scheduled lines

Minru Wang*[1], Patrick Stokkink[2], and Nikolas Geroliminis[1]

[1]Laboratory of Urban Transport Systems, EPFL, Switzerland
[2]Faculty of Technology, Policy, and Management, TU Delft, The Netherlands

Short summary

This paper introduces a multimodal urban parcel delivery framework that integrates drones with scheduled bus services. We formulate the Pick-up-and-Delivery Problem with Drones and Scheduled Lines (PDP-DSL), in which drones serve first- and last-mile legs while parcels may use public transport for middle-mile transfers. The problem is modelled as an arc-based mixed-integer linear program. To address computational complexity and overcome well-known limitations to commercial solvers, this work develops tailored metaheuristics and feasibility checks based on Bellman–Ford for overall feasibility and dynamic programming for battery-related constraints. While a commercial solver drops in solution quality beyond 40 tasks in a synthetic case study, our ALNS with dynamic programming feasibility checks finds good quality solutions for up to 125 tasks. Results further show that in terms of routing distances, the PDP-DSL can be a suitable alternative to truck delivery and drone delivery.

Keywords:
City logistics, drone transport, metaheuristics, pickup and delivery, urban freight transport

1 Introduction

City centres and business districts exhibit topographical characteristics that hinder conventional delivery vehicles. Conventional on-street delivery vehicles create extra congestion during pick-up and drop-off and increase emissions and vehicle-kilometres travelled. We formulate an urban parcel delivery service that leverages spare capacities on existing scheduled bus lines, complemented by a fleet of autonomous drones to cover first- and last-mile segments. This introduces the Pick-up-and-Delivery-Problem with Drones and Scheduled Lines (PDP-DSL). Parcels are picked up at stores, transported by a sequence of drone and bus segments, and finally delivered to customers. We consider both direct drone trips and trips that use a bus line for an intermediate segment, as illustrated in Figure 1.

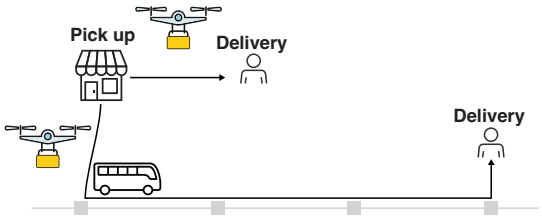


Figure 1: Illustration of direct and multimodal delivery

To minimize operational costs for the delivery platform, we formulate the PDP-DSL problem as a Mixed-Integer Linear Programming (MILP) problem, and obtain in the same model (1) an optimal *parcel path* for every parcel and (2) an optimal *drone path* every (utilized) drone. The model incorporates realistic assumptions such as constant bus and drone velocity, battery swapping, and negligible stopping times.

Literature review

The PDP-DSL problem lies at the intersection of multimodal routing problems involving drones and logistics applications with scheduled lines such as buses and subway lines.
Research on multimodal freight transportation planning and optimization in the urban environment can typically be classified as extensions to Pickup and Delivery Problems (PDPs) (Dumas et al., 1991; Savelsbergh & Sol, 1995). One such extension is the Pickup and Delivery Problem with Time Windows and Scheduled Lines (PDPTW-SL), introduced by Ghilas et al. (2016b). Their work integrates conventional delivery vehicles and public transit lines. Although relevant mathematical models have existed for over twenty years, public transit for freight transport remained mostly in the realms of academic research, and drones have rarely been considered for urban last-mile transportation until a few years ago.
A major advantage of drone-assisted last-mile delivery is the flexibility and accessibility independent of two-dimensional road networks. Therefore, drones are well-suited to overcome shortcomings of conventional delivery trucks, such as mitigating ground congestion and travel time uncertainty. Nevertheless, battery life imposes new operational constraints. Our work models battery swapping operations in a similar fashion as Dorling et al. (2017), proposing a linear battery consumption model with a fixed battery swapping location.
Scheduled public transit lines offer the advantage of high coverage and capacity in passenger transport (Häll et al., 2009; Posada et al., 2017) and freight delivery (Ghilas et al., 2016b,a). This body of literature provides the mathematical foundation for multimodal delivery optimization using bus lines. In the freight context, public transit provides middle-mile transportation while conventional vehicles complement the service with synchronized first- and last-mile coverage.
Our paper aims to exploit the benefits of both scheduled lines and drones, but integrating them within the same last-mile delivery system. We introduce the PDP-DSL through a two-index arc-based formulation, and highlight the following methodological contributions:

- we formulate a multimodal many-to-many last-mile pickup and delivery problem with drones and scheduled lines (PDP-DSL) as an MILP;
- we develop an ALNS solution approach that extends upon the state-of-the-art with problem-specific local operators;

- we introduce an efficient graph-based feasibility check using dynamic programming, instead of computing schedules by propagation, which is commonly done in the literature;

The combination of the latter two demonstrates a clear improvement in terms of computational speed and solution quality over commercial solvers.

2 Methodology

We first give a formal description of the PDP-DSL and a succinct summary of the mathematical formulation (the full formulation is relegated to the full paper). Then, we introduce our meta-heuristic solution algorithm based on ALNS and local search. Finally, we present the dynamic programming-based feasibility check.

Problem description

We consider a system of scheduled bus lines and a fleet of drones serving a set of pickup-delivery requests. The objective is to minimize total drone flight distance, while bus operations are assumed fixed and incur no additional cost. Drones depart from and return to a depot, providing first- and last-mile transport. Parcels may either be delivered directly by drone or transferred via bus stops, requiring temporal synchronization between drone and bus movements. Drone routing is further constrained by limited battery capacity, allowing swaps only at designated locations.

Assumptions Drones carry at most one parcel (capacity Q=1), while buses have unlimited capacity. Bus services are high-frequency and modelled like conveyor belts. Parcels may transfer only within a single bus line, and loading/unloading occurs at stops. Battery swaps are allowed only when drones are empty. For simplicity, but without loss of generality, service and dwell times are neglected.

Mathematical formulation The PDP-DSL is defined on a directed graph $G = (N, A)$, with node set $N = O^{orig} \cup N^p \cup N^d \cup T \cup O^{dest}$, and arc set $A = N \times N$. In the node set, O^{orig} and O^{dest} represent sets of depot nodes replicated by the number of drones m, respectively for drones leaving and returning to the depot. Let N^p denote the set of pickup nodes and N^d the set of delivery nodes. T denotes the set of bus stops replicated by the number of tasks n. Location R is the battery swapping station where a drone can acquire a new battery before the current one is depleted; this location is not included in the node set N.
Drone flight distance d_{ij} along each arc $(i, j) \in A$ is the Euclidean distance between nodes i and j. For battery swapping, the drone flight distance between station R and node $i \in N$ is d^*_i. The flight distance from node i to j passing through swap station R is denoted as $d^*_i + d^*_j = d^*_{ij}$. Drones fly at an average velocity v_d. Bus travel time t^{bus}_{ij} along a bus arc (i, j) between adjacent stops i and j is preprocessed by assuming constant bus speed v_b over the distance between stops $i, j \in T$. Assuming linear drone battery consumption, the battery level of a fully charged battery is set to B with its unit in distance; binary variable $\sigma_{ij} = 1$ if a battery swap takes place along arc $(i, j) \in A$. The objective function of the PDP-DSL is as follows:

$$\min \sum_{i,j \in N} d_{ij} x^{drone}_{ij} + (d^*_{ij} - d_{ij})\sigma_{ij} \tag{1}$$

We formulate an arc-based mixed-integer program with standard flow conservation constraints for drones and parcels, travel time propagation constraints, synchronization constraints at pickup, transfer, and delivery nodes, and battery dynamics along drone routes.
Explicitly solving this formulation is computationally challenging due to the combinatorial interaction between routing, synchronization, and battery usage. The feasibility of the resulting MILP is constrained by two main factors. First, time consistency must be maintained along both drone and parcel paths, including synchronization between drones and buses at bus stops serving as transfer points. Second, the total flight distance between consecutive battery swaps cannot exceed a fixed range, and battery swaps can only occur when the drone is not carrying a parcel. These constraints induce strong coupling between routing, timing, and battery use.
Instead of relying on commercial solvers to tackle the full formulation, we exploit its structure to design a dynamic programming algorithm that efficiently checks the feasibility of candidate routes, as described next.

Adaptive large neighbourhood search

Our model validation phase highlighted two challenges commercial solvers face in solving the PDP-DSL: firstly, in obtaining an initial solution, and secondly, in improving incumbent solutions in MILPs with many variables and constraints. We develop a metaheuristic that can leverage the combinatorial nature to systematically explore neighbourhoods in the solution space and, with that, move towards improving solutions. Specifically, we propose an ALNS algorithm (as first introduced by Ropke & Pisinger (2006)) and implement it using the Python package by Wouda & Lan (2023).

A feasible initial solution is obtained by serving all tasks with direct drone flights. The initial solution is then iteratively altered using a combination of destroy and repair operators.

Destroy operators select tasks in drone paths that meet the removal criteria of each operator. We implement three operators that have demonstrated usefulness in Ropke & Pisinger (2006) and Ghilas et al. (2016a). These operators are *random removal, worst distance removal*, and *longest distance removal*.

Tasks previously removed by a destroy operator are inserted into the drone routes using one of eight variants of repair operators that combine two insertion operators with three local search operators. Based on Ropke & Pisinger (2006); Ghilas et al. (2016a) we deploy *Greedy Insertion (GI)* and *λ-Feasible Insertion (λFI)*. We develop three *local search operators* specific to the PDP-DSL that identify further improvements to the obtained solution. The local search operators (i) switch two bus stops on the same drone path, (ii) choose an alternative bus stop on a drone path, (iii) combine (i) and (ii). Applying these after one of the repair operators leads to eight combinations.

All operators start with equal selection probability, and their scores are updated each iteration using a segmented roulette wheel as implemented in the ALNS package by Wouda & Lan (2023). A simulated annealing acceptance criterion is applied at each iteration, with temperature parameters set via the autofit method of Wouda & Lan (2023).

Although commercial solvers can be directly used to check the feasibility and evaluate objective function values of ALNS incumbent solutions for up to 10 tasks, this is computationally intractable for larger instances. The remainder of this section proposes metaheuristics for the PDP-DSL with custom feasibility checks.

Feasibility check using Bellman-Ford and recursion

During each ALNS iteration, new solutions are evaluated for feasibility, and infeasible solutions are discarded. The most cumbersome constraint to verify is that arising from drone-bus synchronization: when a parcel is transported by bus, the assigned drone must arrive at the destination stop no later than the parcel. Given that direct schedule propagation becomes cumbersome for long routes, we exploit the Bellman-Ford algorithm (Cormen et al., 2009). Given a candidate solution defined by drone routes and parcel arcs, we assess feasibility using a combination of Bellman-Ford (for general feasibility) and dynamic programming (for adding battery swapping arcs). Algorithm 1 describes the proposed two-stage feasibility check. Details on both stages are provided in the next subsections.

The transformation in Line 2 transforms G_1 into a single-source graph $G_2 = (N_2, A_2)$ on which the Bellman-Ford algorithm is applied. A negative weight cycle in G_2 indicates that even if drones have unlimited battery, the drone fleet is unable to travel along the assigned path and fulfill all pick-up and delivery schedules; consequently, the set of drone paths is infeasible. For infeasible solutions, a battery swap is incorporated through dynamic programming, followed by a final run of Bellman-Ford to evaluate the feasibility of the new route.

Bellman-Ford feasibility check

A feasible schedule can be represented with a system of difference constraints containing: (1) equality constraints between the end and start nodes for parcels travelling along bus arcs, and (2) greater than or equal to constraints between the end and start nodes for arcs traversed by drones. The former requires strict synchronization, so that a parcel is dropped off on a bus and picked up from a bus at the exact moment of its scheduled arrival. In its current form, our approach only evaluates one battery swapping arc. If a solution is infeasible, the algorithm does not check for the next best solution, even if performing the swap elsewhere may produce a feasible solution. This could be incorporated through an iterative algorithm, which is marked as a future research direction.

Algorithm 1: Two-stage feasibility check for drone paths

Input: Paths of m drones $P = \{P_0, \ldots, P_{m-1}\}$, drone travel times t_{ij}^{drone} between nodes i, j, bus travel times t_{ij}^{bus}

Output: feasibility check outcome, objective function value if feasible

```
Construct G_1 = (N_1, A_1) from drone paths P and travel times;
Transform G_1 into canonical form G_2, and run Bellman-Ford on G_2;
if Negative weight cycle then
    solution infeasible;
else
    if All drone paths shorter than B then
        solution feasible;
    else
        Optimize battery swap with dynamic programming;
        Update battery swap arc weights in G_2 and run Bellman-Ford;
        if Negative weight cycle then
            solution infeasible;
        else
            solution feasible;
```

Figure 2 provides an illustrative example of two parcels that use paths consisting of two drones and a bus. Stars mark the drone depots, triangles mark pickup points for parcels, circles mark dropoff points for parcels, and squares mark bus stops. Here, a solution is feasible only if both drones are able to unload the parcels by arriving at designated unloading bus stops before the parcel on the bus does. In this example, this suggests that the solution is only feasible if $w_{8,9} \leq 5$.

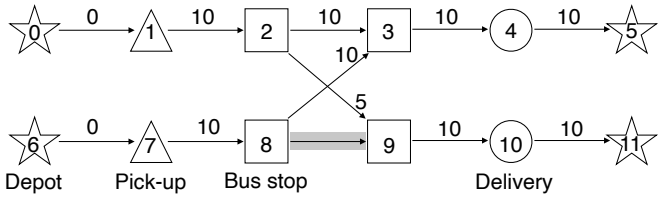


Figure 2: Illustration of $G_1 = (N_1, A_1)$ with drone or bus travel times along weighted arcs. Feasible only if $w_{8,9} \leq 5$.

The following provides an illustrative example for detecting arc infeasibility. Drone flight times and bus travel times are labelled as arc weights in Figure 2, except for that of arc $(8, 9)$ shaded in grey which is unknown, as its weight can change the feasibility of the problem. Each drone must arrive at a bus stop before the parcel on the bus does; one can verify that the solution is only feasible if $w_{8,9} \leq 5$ holds.

Although trial-and-error easily detects an infeasibility in this trivial example, it becomes impractical as the graph size increases. We overcome this obstacle with a system of linear inequalities for feasibility check. Let τ_i be the arrival time of a drone at node $i \in N_1$. Let A_1^{bus} be the set of arcs travelled by parcels on buses, and A_1^{drone} the set of arcs travelled by drones. The system of difference constraints in (2) expresses travel times along drone arcs, and (3) for travel times on bus arcs, as well as non-negativity constraint in (4) for departure times from depot nodes.

$$\tau_j \geq \tau_i + w_{ij} \quad \forall (i,j) \in A_1^{drone} \tag{2}$$

$$\tau_j = \tau_i + w_{ij} \quad \forall (i,j) \in A_1^{bus} \tag{3}$$

$$\tau_i \geq 0 \quad \forall i \in \{0, 6\} \tag{4}$$

We complete the formulation with objective (5) of minimizing the sum of the drone arrival times, which corresponds to finding the earliest feasible schedule:

$$\min \sum_{i \in N_1} \tau_i \tag{5}$$

To apply the Bellman-Ford algorithm, the final step is to match its canonical form as presented in Cormen et al. (2009) with a maximization objective of $\max -\sum_{i \in N_1} \tau_i$ with inequalities rearranged. Chapter 24.4 of Cormen et al. (2009) provides a formal proof that a system of linear inequalities has a feasible solution if its corresponding constraint graph contains no negative-weight cycles.

Dynamic programming for battery swap optimization

The battery-swapping problem selects swap points to ensure constraints of the battery life of a drone are satisfied, while still minimizing its total travel distance. This procedure is started when no feasible solution exists without battery swaps, as marked in Algorithm 1. Along a long route, battery swapping may repeatedly take place, thus leading to a natural structure of overlapping subproblems. This structure lends itself perfectly for dynamic programming. The full algorithm is omitted due to space constraints and will be included in the full version of the paper.

3 Results and discussion

This section first provides visualizations of drone and parcels paths as well as battery levels for a small problem instance. Then, we evaluate the performance of the ALNS algorithm compared to Gurobi and show the scalability of the algorithm. Finally, we compare the PDP-DSL to traditional and drone routing problems.

Drone and parcel path visualization

Figure 3 provides a visualization for the optimal solution to a small instance with $n = 10$ tasks randomly generated in a 10 km by 10 km area, 1 bidirectional bus line with $s = 10$ stops located 1 km apart, a depot, and a battery station located near the centre. The number of drones is $m = 2$. The maximum battery level B is set to 20 km (thereby implicitly enforcing the use of the battery swap station). Under these settings, parcels are delivered through a combination of direct parcel paths and indirect parcel paths using the bus lines.

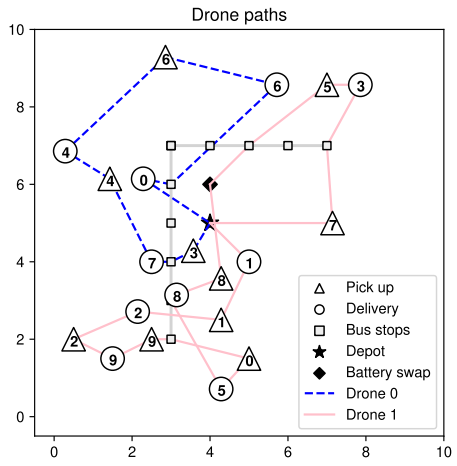
Comparison between Gurobi and ALNS

Table 1 compares the performance of Gurobi with ALNS for 10 to 40 tasks. The drone fleet size m is provided as an input parameter; preliminary experiments indicate that, while larger fleet sizes increases solution symmetry, its impact on computational performance is limited compared to the number of tasks. Both methods are terminated upon reaching a 1% optimality gap or a computational limit; for larger instances where this is not achieved, the best incumbent solutions are reported. Both approaches can find improving feasible solutions, even though this takes much longer for a commercial Gurobi solver, frequently encountering memory issues. The performance of the metaheuristic approach on the same instances are also presented in Figure 4, with ALNS destroy fraction $\gamma = 0.5$. Our study finds that Gurobi tends to plateau for a long time without finding improving solutions, whereas ALNS is able to efficiently explore the solution space and improve the best solution. The times taken for ALNS with $300 - 500$ iterations are all clear improvements compared to the Gurobi performances, even though good solutions are already found before the end of the iteration limit. Nevertheless, given that the computational time increases rapidly with the number of tasks, a promising next step can consider the parallelization of insertion operators, or the elimination of low-quality insertions up front.

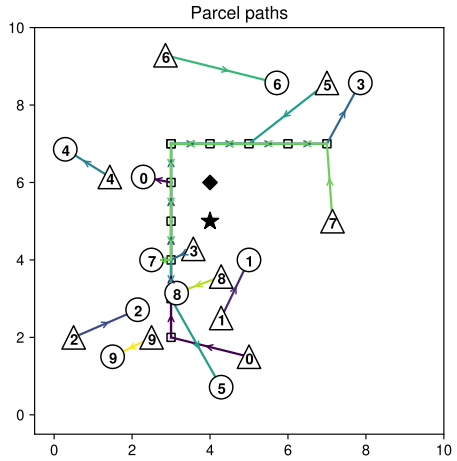
Since the ALNS always begins with a feasible initial solution that is easy to construct, this serves as a clear advantage over Gurobi, which requires non-negligible MILP model construction time prior to solving it. Figure 5 presents results for 50, 75, and 125 tasks over 200 iterations. The pickup and delivery locations of the tasks are randomly generated, and the same bus line in Figure 3 is used. Despite an increase in problem size, the metaheuristic approach remains efficient at exploring the solution space to obtain improvements.

Comparison with truck delivery

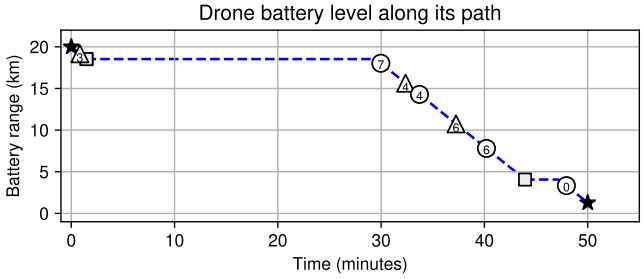
Finally, we quantify the benefit of using drones and scheduled lines in a last-mile delivery setting. Table 2 displays the total vehicle/drone routing distance for three configurations and for different



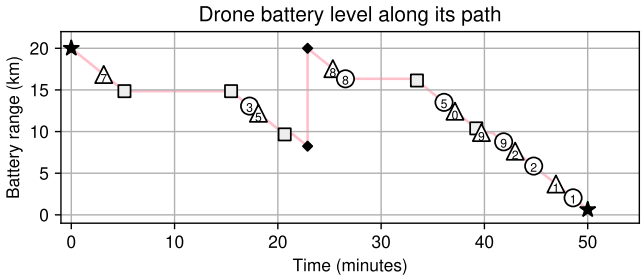
(a) Optimal paths of 2 drones



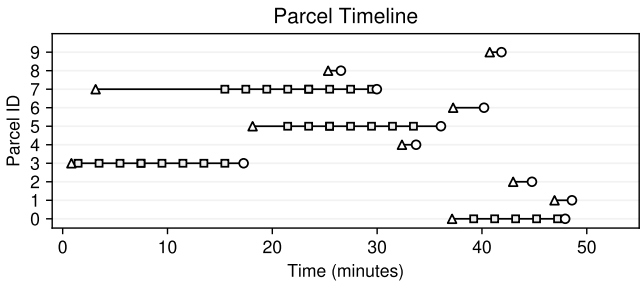
(b) Optimal parcel paths



(c) Timeline of drone 0



(d) Timeline of drone 1



(e) Parcel departure times

Figure 3: Optimal drone and parcel routing for 10 tasks, 2 drones, 1 bus line

Table 1: Comparison for Gurobi and ALNS, for 10-40 tasks (n) and m drones. The theoretical lower bound (LB) is provided by Gurobi. OF denotes the Objective Function when the algorithm terminates. The Gurobi OF is the incumbent at termination. The optimality gap is computed as the percentage difference between OF and LB. The computation time for both algorithms is denoted in seconds. The method in **bold** indicates the option with the lower objective.

		Gurobi				ALNS		
n	m	Theoretical LB (km)	OF (km)	Gap	Time (s)	OF (km)	Gap	Time (s)
10	2	73	**74**	1%	222981	**74**	1.0%	42
20	3	107	**123**	12.60%	345487	126	17.7%	476
30	3	149	196	24.10%	24001	**181**	21.9%	14882
40	3	193	253	23.80%	344307	**237**	22.8%	20205

instance sizes. The VRP with a capacity of 1 is used as a proxy for drone delivery and the truck delivery counterpart is constructed as a VRP with a capacity of 150. The results indicate that including scheduled lines has a clear benefit over drone-only delivery. Nevertheless, traditional routing remains unmatched in terms of routing efficiency. Clearly, the specific routing costs are dependent on the network configuration as well as mode-specific cost parameters, for which a sensitivity analysis will be included in the full paper.

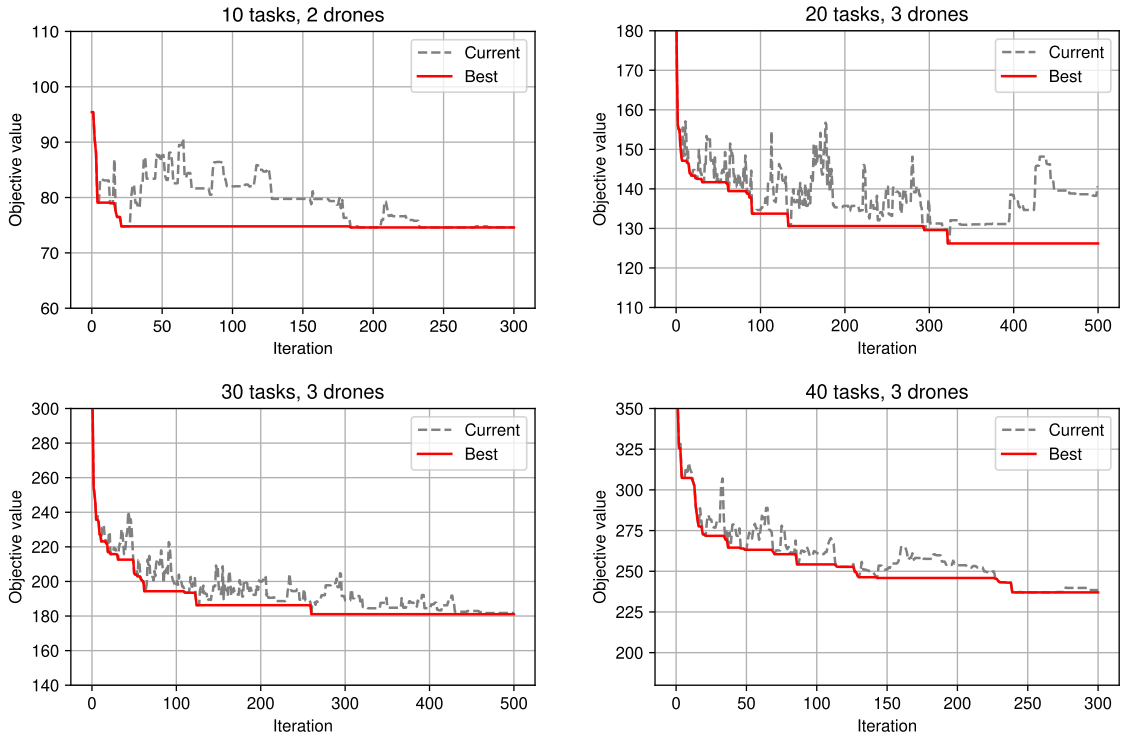


Figure 4: ALNS performance on the same instances as in Table 1

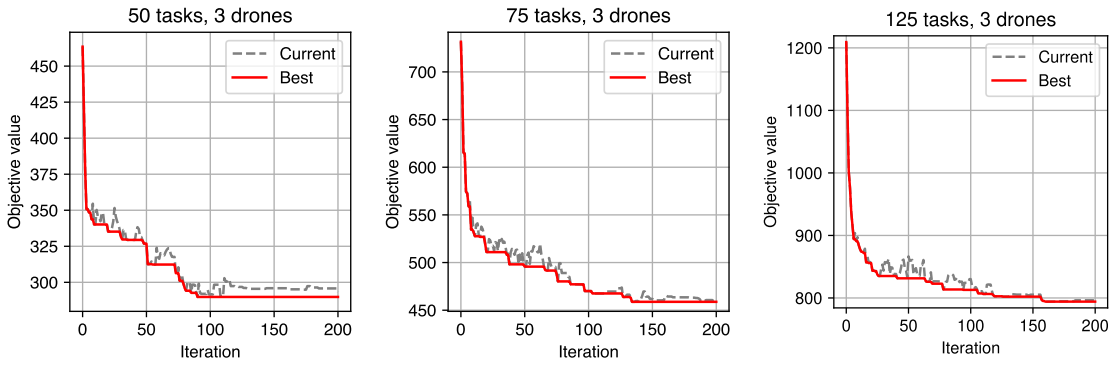


Figure 5: ALNS results for 3 drones and removal fraction $\gamma = 0.2$ for 50, 75, and 125 tasks

Table 2: Total vehicle/drone routing distances in km for a single truck with capacity $Q = 1$ (row 1), $Q = 150$ (row 2), and drone with capacity 1 (row 3).

n	**10**	**20**	**30**	**40**	**50**	**75**	**125**
VRP (Q=1)	102	164	276	347	372	597	1013
VRP (Q=150)	63	76	87	113	123	159	232
PDP-DSL	74	126	181	237	289	461	790

4 Conclusion and future work

In order to assess the viability of an innovative multimodal last-mile delivery system using drones and buses, this paper presents our metaheuristic approach with graph-based feasibility checks suitable for solving the problem at a large scale. In this work, synchronization constraints between the bus and drone increase the modelling complexity. Although Gurobi can serve as a reliable tool for early stage validation for MILPs such as the PDP-DSL in small instances up to 20 tasks, its computational cost constitutes a bottleneck for large scale implementation.
In contrast with commercial MILP solvers, we have shown that the proposed ALNS framework performs well both in terms of solution quality (matching Gurobi performance for 10 to 40 tasks)

and speed (finding improving solutions in under 200 iterations within reasonable time). Furthermore, the ALNS shows promising performance for up to 125 tasks and 3 drones. Nevertheless, the algorithm can benefit from insertion parallelization to reduce computation time. Other promising directions can consider additional ALNS operators as well as parameter tuning.
For a real-life implementation of the PDP-DSL last-mile delivery service, practitioners can model operational challenges associated with loading and unloading delays, parcel integrity, user safety, etc. Future investigations can incorporate requirements related to drone battery management, capacity constraints, handling of parcels at bus stops, and travel time uncertainties.

References

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms, third edition* (3rd ed.). The MIT Press.

Dorling, K., Heinrichs, J., Messier, G. G., & Magierowski, S. (2017, January). Vehicle Routing Problems for Drone Delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *47*(1), 70–85. doi: 10.1109/TSMC.2016.2582745

Dumas, Y., Desrosiers, J., & Soumis, F. (1991, September). The pickup and delivery problem with time windows. *European Journal of Operational Research*, *54*(1), 7–22. doi: 10.1016/0377-2217(91)90319-Q

Ghilas, V., Demir, E., & Van Woensel, T. (2016a, August). An adaptive large neighborhood search heuristic for the Pickup and Delivery Problem with Time Windows and Scheduled Lines. *Computers & Operations Research*, *72*, 12–30. doi: 10.1016/j.cor.2016.01.018

Ghilas, V., Demir, E., & Van Woensel, T. (2016b, April). The pickup and delivery problem with time windows and scheduled lines. *INFOR: Information Systems and Operational Research*, *54*(2), 147–167. doi: 10.1080/03155986.2016.1166793

Häll, C. H., Andersson, H., Lundgren, J. T., & Värbrand, P. (2009, May). The Integrated Dial-a-Ride Problem. *Public Transport*, *1*(1), 39–54. doi: 10.1007/s12469-008-0006-1

Posada, M., Andersson, H., & Häll, C. H. (2017, July). The integrated dial-a-ride problem with timetabled fixed route service. *Public Transport*, *9*(1), 217–241. doi: 10.1007/s12469-016-0128-9

Ropke, S., & Pisinger, D. (2006, November). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4), 455–472. doi: 10.1287/trsc.1050.0135

Savelsbergh, M. W. P., & Sol, M. (1995, February). The General Pickup and Delivery Problem. *Transportation Science*, *29*(1), 17–29. doi: 10.1287/trsc.29.1.17

Wouda, N. A., & Lan, L. (2023). ALNS: a Python implementation of the adaptive large neighbourhood search metaheuristic. *Journal of Open Source Software*, *8*(81), 5028. doi: 10.21105/joss.05028