Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2021) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

The 12th International Conference on Ambient Systems, Networks and Technologies (ANT)
March 23 - 26, 2021, Warsaw, Poland

Delivery drone route planning over a battery swapping network

Taner Cokyasar*

Argonne National Laboratory, Energy Systems Division, 9700 S. Cass Ave., Lemont, IL, 60439, USA

Abstract

Many enterprises invest on drone delivery research and development to drop off packages at consumers' doorsteps in a matter of minutes. We study delivery drone route planning over a battery swapping network allowing farther reach by penetrating current battery capacity constraints. A mixed-integer nonlinear programming model is created to plan efficient drone routing over the swapping machines by minimizing the delivery lead time. We develop an exact solution method, evaluate its performance, and compare it with a straightforward nonlinear solver application. A case study highlights the applicability of the model. Data and source code to the solver are publicly shared.

Keywords: Drone delivery; drone routing; drone battery swapping; mixed-integer nonlinear programming; optimization

1. Introduction

Last-mile drone delivery technology is in the development phase and a large-scale deployment is afar off. Moreover, how the deployment will be made is also ambiguous. Practitioners and researchers have proposed possible last-mile drone delivery deployment scenarios, such as using drones as an assistive technology to the current truck deliveries [13, 14], re-charging drone batteries above public transport and rail [10, 11], building bee-hive-shaped fulfilment centers (FCs) to deliver customers within 30 minutes of flight reach [6], and constructing a delivery network with automated battery swapping machines (ABSMs) to allow drones fly longer distances [3, 4, 5, 7, 9, 15]. These studies focused on the long-term decision-making aspect of the problem. In other words, they proposed mathematical models to determine where ABSMs should be located to minimize the long-term system costs, including battery swapping machines, drones, batteries, and so on.

In this study, we take the aforementioned studies into account and solve the delivery drone route planning (DDRP) problem by assuming the ABSM location decision has already been made. Medium-term planning is useful in cases, where the operational-level problem is computationally complex and expensive to solve. We contribute to the literature

* Corresponding author. Tel.: +1-630-252-1060.
E-mail address: tcokyasar@anl.gov

by formulating a model for the DDRP, developing an exact solution method, and illustrating a realistic case study to support the applicability of our approach. We also contribute to practice by proposing a solution to a problem that is likely to arise in the near future.

In the following sections, we provide a brief literature review, model the problem with an MINLP, and develop a cutting-plane-based exact solution approach requiring to iteratively solve a series of mixed-integer quadratically-constrained programming (MIQCP) problems. The goal of our model is to route drones from an FC to a set of customer locations while minimizing the last-mile delivery lead time, including the travel time and the ABSM queueing time, over a medium-term planning horizon. Following the model and solution method, we present computational results showing the capabilities of our model and a case study illustrating its applicability in real-world problems. To support transparency in research and to attract future research in the subject, we publicly share the data used and the source code to our solver at

https://gitlab.com/tcokyasar/ddrp-optimization.

2. Literature Review

Hong et al. [9] developed a mixed-integer programming (MIP) model to design a drone delivery network with ABSMs by maximizing the total delivery coverage. Cokyasar et al. [4] further investigated the problem and developed a mixed-integer nonlinear programming (MINLP) model minimizing the total delivery cost over time. They considered the cost of waiting time (to receive the battery swapping service), congestion at ABSMs, and the optimal partition of demand into drones and trucks while making the ABSM location decisions. In their study, the long-term routing was one-way meaning that drones need to take the same route while going to a delivery and returning from the delivery. To address this issue, Cokyasar et al. [5] developed another MINLP that consider round-trip routing and the optimal number of drones. In that study, battery availability was not taken into account by assuming ABSMs would always keep an abundant fully-charged battery inventory. Cokyasar [3] focused on this aspect and proposed a considerably different queueing approach to find the optimal number of batteries to be kept at each ABSM location.

A drawback–if one needs to be found–in the above studies was the consideration of a single FC in the long-term decision-making. Shavarani et al. [15] built an MINLP that considers the decision-making for multiple FCs. As a result of increase in the model complexity, finding an exact solution to the problem became challenging. Hence, they solved the problem using a meta-heuristic algorithm with no guarantee to the optimal solution. Since the actual deployment scenario is still an unknown, focusing on a single FC can be considered acceptable and benefits a pilot deployment.

3. Model

The DDRP can be described as follows. A single FC (denoted by the set notation $\{0\}$) and a set of ABSMs (denoted by $\mathcal{I}$) allow delivering parcels by drones to a set of customers (denoted by $\mathcal{D}$). The delivery network can be shown with the graph notation as $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{0\} \cup \mathcal{I} \cup \mathcal{D}$ is the set of vertices and $\mathcal{E} = \{(i, j) | i, j \in \mathcal{V} \wedge i \neq j \wedge T_{ij} \leq R\}$ denotes the set of edges that can be traveled by drones without a need for battery swapping. Here, T_{ij} is the travel time on any origin-destination pair (i, j), where $T_{ij} \in \mathbb{R}^+$, and R is the maximum travel time without battery swapping. In this setting, once the order arrives to the FC, a drone is loaded with one unit, and it is delivered to the customer location by traveling along the feasible edges defined in $\mathcal{E}$. The delivery drone's battery is automatically swapped with a fully-charged battery at ABSM locations on its path to the customer. The battery swapping time is a constant denoted by $1/\mu$. The DDRP problem is to find the optimal drone routing with battery swapping at ABSMs to minimize the customer waiting time (i.e., delivery lead time) for all customers over a planning horizon. The delivery lead time includes the travel time spent and an average queueing time waited at ABSM locations. Since order arrivals are often stochastic and a constant time is needed to swap batteries, drones delivering to different customers are likely to come across and form a queue to receive battery swapping service at ABSM locations. Table 1 includes notations used in the model.

Before providing the model formulation, we further elaborate how the delivery network functions. The demand follows a Poisson process with demand rate λ_d for customer $d \in \mathcal{D}$. As the current drone delivery systems focus on a single package delivery, we assume that one unit is demanded at a time, and a drone can deliver one unit at a time.

Table 1: Set, parameter, and variable definitions.

Set	*Definition*
$\mathcal{V}$	set of vertices, indexed by $i \in \mathcal{V} = \{0\} \cup \mathcal{I} \cup \mathcal{D}$.
$\mathcal{D}$	subset of vertices for demand locations, indexed by $d \in \mathcal{D}$ and $\mathcal{D} \subset \mathcal{V}$.
$\mathcal{I}$	subset of vertices for battery swapping machine candidate locations, indexed by $i \in \mathcal{I}$ and $\mathcal{I} \subset \mathcal{V}$.
$\mathcal{E}$	set of edges, indexed by the tuple $(i, j) \in \mathcal{E} = \{(i, j) \mid i \in \mathcal{V} \setminus \mathcal{D} \wedge j \in \mathcal{V} \setminus \{0\} \wedge i \neq j \wedge T_{ij} \leq R\}$.
$\mathcal{I}_j(\{\cdot\})$	subset of feasible origin vertices of a given vertex $j \in \mathcal{I}$ that can be traveled from without battery swapping, $\mathcal{I}_j(\{\cdot\}) = \{i \mid i \in \{\cdot\} \wedge T_{ij} \leq R \wedge i \neq j\}$, where $\{\cdot\}$ represents the input set for index i.
Parameter	*Definition*
T_{ij}	travel time on any origin-destination pair (i, j), where $i, j \in \mathcal{V}$.
λ_d	average demand rate at vertex $d \in \mathcal{D}$ during the planning horizon, i.e., average number of units demanded by $d \in \mathcal{D}$ over time.
μ	service rate of battery swapping machine during the planning horizon.
R	maximum travel time without battery swapping.
Variable	*Definition*
y_{ijd}	= 1, if a drone flies on edge $(i, j) \in \mathcal{E}$ to deliver the demand at vertex $d \in \mathcal{D}$; 0, otherwise.
ρ_j	utilization rate at vertex $j \in \mathcal{I}$.

Every location $j \in \mathcal{I}$ houses a single ABSM, and these machines do not carry any inventory since they are designed to occupy a small space and allow drones to fly longer distances. We assume drones are always available and ready-to-fly equipped with fully-charged batteries at the FC. Drones fly at a constant speed, hence T_{ij} can be estimated by Euclidean distance between two vertices. We plan the routing from FC to customers and assume the return routes are the same. A *route*, in this context, is defined as a flight on any given edge $(i, j) \in \mathcal{E}$, and a *path* is a combination of routes with the initial origin of the FC and the final destination of a customer location. We assume that ABSMs always carry a fully-charged battery inventory at the time of a swapping. Note that most of these assumptions are acceptable in a *medium-term planning* layout, yet an operational decision-making model should relax some of these assumptions (e.g., battery availability at ABSMs, drone availability at the FC, etc.) to construct a fully-functional delivery system.

We present the following MINLP model to solve the DDRP problem.

$$\textit{Minimize } \mathbf{T} = \sum_{(i,j)\in\mathcal{E}, d\in\mathcal{D}} \lambda_d T_{ij} y_{ijd} + \sum_{(i,j)\in\mathcal{E}\wedge j\notin\mathcal{D}, d\in\mathcal{D}} \lambda_d \left[\frac{1}{\mu} + \frac{\rho_j}{2\mu\left(1-\rho_j\right)}\right] y_{ijd}, \tag{1}$$

$$\sum_{i\in\{i|i\in\mathcal{I}\wedge T_{id}\leq R\}} y_{idd} = 1 \qquad \forall d \in \mathcal{D}, \tag{2}$$

$$2T_{id} y_{idd} \leq R \qquad \forall i \in \{i|i \in \mathcal{I} \wedge T_{id} \leq R\}, d \in \mathcal{D}, \tag{3}$$

$$\sum_{\mathcal{I}_j(\mathcal{V}\setminus\mathcal{D})} y_{ijd} = \sum_{i\in\mathcal{I}_j(\mathcal{V}\setminus\{0\})} y_{jid} \qquad \forall j \in \mathcal{I}, d \in \mathcal{D}, \tag{4}$$

$$\rho_j = \frac{1}{\mu} \sum_{i\in\mathcal{I}_j(\mathcal{V}\setminus\mathcal{D}), d\in\mathcal{D}} 2\lambda_d y_{ijd} \qquad \forall j \in \mathcal{I}, \tag{5}$$

$$y_{ijd} \in \{0, 1\},\ \rho_j \in [0, 1)\,. \tag{6}$$

The objective function (1) minimizes the total travel time (from FC to customers) for all deliveries and the average queueing time (including the battery swapping service time) over a planning horizon. The first component (Σ) corresponds to the travel time for drones in product with the demand rate. In the second component, $\rho_j / \left[2\mu\left(1-\rho_j\right)\right]$ is the average queue waiting time for a drone at a given ABSM $j \in \mathcal{I}$, and this term is denoted by W_j^q in the $M/D/1$ queueing system. Model formulations and explanations for this queue system can be found in many books, such as

Kleinrock [12, Chapter 5, pp. 188]. Adding the service time $(1/\mu)$ to W_j^q yields the overall waiting time (denoted by W_j) to receive the battery swapping service at ABSM $j \in \mathcal{J}$. This term is multiplied by λ_d to account for the overall queueing time for all customers' deliveries over a planning horizon. Constraint sets (2)–(4) satisfy the connectivity of edges. In (2), we enforce that all deliveries are carried out by drones. Here, the first index is all ABSM locations (origins) that can be traveled from a given a customer $d \in \mathcal{D}$, the second index is the destination, and the third index d points to the customer. Constraint set (3) satisfies that a drone flying on edge (i, d) would have adequate battery to return to the same ABSM. Constraint set (4) ensures a drone entering to a vertex leaves the vertex. Constraint set (5) defines the variable ρ_j in terms of the routing variable $\mathbf{y}$. Essentially, $\rho_j = \lambda_j/\mu$, where $\lambda_j = \sum_{i \in \mathcal{I}_j(\mathcal{V}\setminus\mathcal{D}), d\in\mathcal{D}} 2\lambda_d y_{ijd}$, and the constant '2' accounts for the drones returning back to the FC after a delivery to a customer. Although routing is planned only from the FC to customers, average queue wait times are impacted by (unloaded) drones returning to the FC. Note that (5) is redundant as the right-hand-side can be plugged into its correspondence in the objective function, however, we prefer to separate it as an equality constraint for a more tractable presentation. Lastly, (6) includes the non-negativity conditions and variable domains.

The program is nonlinear due to $\rho_j y_{ijd}/(1-\rho_j)$ term in (1). Although solvers, e.g., Gurobi, IBM CPLEX, Ipopt, etc., may successfully handle these quadratic expressions, the computational performance is not always as desired. The following section is devoted to developing an efficient algorithm to optimally solve the problem.

4. Solution Method

We define $W_j^q = \rho_j / \left[2\mu\left(1-\rho_j\right)\right]$, where $W_j^q \in \mathbb{R}^+$, and replace its correspondence in (1). Therefore, the second component in the objective function becomes $\sum_{(i,j)\in\mathcal{E}\wedge j\notin\mathcal{D}, d\in\mathcal{D}} \lambda_d \left[\frac{1}{\mu} + W_j^q\right] y_{ijd}$. To address the nonlinearity due to $W_j^q y_{ijd}$, we can introduce a variable, $\beta_{ijd} = W_j^q y_{ijd}$, where $\beta_{ijd} \in \mathbb{R}^+$, and we present the linearized objective as follows.

$$\textit{Minimize } \mathbf{T} = \sum_{(i,j)\in\mathcal{E}, d\in\mathcal{D}} \lambda_d T_{ij} y_{ijd} + \sum_{(i,j)\in\mathcal{E}\wedge j\notin\mathcal{D}, d\in\mathcal{D}} \lambda_d \left(\frac{y_{ijd}}{\mu} + \beta_{ijd}\right), \tag{7}$$

After this linearization, we should also constrain the model to reflect $\beta_{ijd} = W_j^q y_{ijd}$ and $W_j^q = \rho_j / \left[2\mu\left(1-\rho_j\right)\right]$. Although the first one can be easily addressed by introducing three additional constraint sets, using this quadratic expression in a constraint and solving the problem in an MIQCP framework does not cause a considerable performance decrease in commercially available solvers. (Indeed, some solvers' algorithms can better deal with this expression in many cases.) In other words, we can introduce the following quadratic constraint and let solvers deal with it.

$$\beta_{ijd} = W_j^q y_{ijd} \qquad \forall (i,j) \in \mathcal{E} \wedge j \notin \mathcal{D}, d \in \mathcal{D} \tag{8}$$

At this point, we can solve the resulting MIQCP (2)–(8) with an additional constraint for $W_j^q = \rho_j / \left[2\mu\left(1-\rho_j\right)\right]$. However, the convexity proof provided by [5] allows us to introduce a cutting-plane constraint along with an algorithm to *potentially* solve the problem faster. Briefly, W_j^q is convex in ρ_j. This means W_j^q is supported by $a + b\rho_j$ at $\rho_j = \rho'_j$, where $a = \rho'_j / \left[2\mu\left(1-\rho'_j\right)\right] - b\rho'_j$ and $b = \frac{\partial\left[\rho'_j/\left(2\mu\left(1-\rho'_j\right)\right)\right]}{\partial\rho'_j}$. Therefore, $W_j^q \geq a + b\rho_j$ is a valid cut for any $j \in \mathcal{J}$, and we introduce it as follows.

$$W_{j'}^q \geq a_{jn} + b_{jn}\rho_{j'} \qquad \forall\, j, j' \in \mathcal{J} \tag{9}$$

In (9), a_{jn} and b_{jn} are constants obtained by ρ_j value of iteration n. For instance, the program formed by (2)–(8) can be solved at iteration 0. Then, (9) can be added at iteration 1 based on the values of ρ_j at the end of iteration 0. This process can be repeated until convergence is achieved. At the end of every iteration, we obtain a lower bound $(\mathbf{T}_n)$, which is provided by the solver, and an upper bound $(\widehat{\mathbf{T}}_n)$ that can be calculated by finding the value of W_j^q by plugging in ρ_j values obtained. Therefore, once the overall gap between $\mathbf{T}_n$ and $\widehat{\mathbf{T}}_n$ approaches to 0 or a satisfactorily low optimality gap value (Δ), the convergence is achieved. In other words, convergence is achieved by satisfying: $\left(\widehat{\mathbf{T}}_n - \mathbf{T}_n\right)/\widehat{\mathbf{T}}_n < \Delta$, where $\Delta \to 0$, and $\left(\widehat{\mathbf{T}}_n - \mathbf{T}_n\right)/\widehat{\mathbf{T}}_n$ is defined as the *gap*.

We can also introduce (9) as a pre-defined user-cut for a set of constant values of ρ_j at iteration 0 to help the solver. For instance, a and b can be calculated based on a set of values in the range $(0, 1)$ with a step size of 0.1, and (9) can be added in addition to (2)–(8) based on these constants at iteration 0. Following the iterative procedure will guarantee the global optimality.

5. Numerical Experiments

In this section, we illustrate computational results indicating the capabilities of our solution method. We solve a set of instances with and without cut constraints and compare the computational efficiency. Further, we illustrate a case study implemented on a portion of the Chicago metropolitan area.

First, we explain the parameter settings used in these experiments. The battery swapping time $(1/\mu)$ is reportedly 3 minutes [1]. Travel time between locations (T_{ij}) is estimated by the Euclidean distance. The demand (λ_d) is calculated by the product of population and a daily demand estimate of 1.051737×10^{-2} per person [15] in a selected demand zone. The demand zones and their corresponding populations are obtained from POLARIS–an agent based simulation assessing the impact of future mobility technologies [2]. We used $R = 10$ minutes by considering a constant drone speed of 60 mph. Finally, all times are converted to daily units, all computations were carried out on an Intel® Core i9 CPU@2.3 GHz MacBook Pro with 32GB of RAM, and problem instances were solved using the Python 3.7 interface to the commercially available solver Gurobi 9.0.2 [8].

5.1. Computational results

We carried out computational experiments for different problem sizes and a number of instances by randomly generating different problems each of which is called an *instance*. Unless otherwise stated, the computational time was capped by one hour and $\Delta = 5.0 \times 10^{-4}$ for each instance. To solve instances, we used (2)–(8) supported by iterative additions of (9) and pre-defined user-cuts, given a set of ρ_j estimates in the range $(0, 1)$ with a step size of 0.1. To compare the performance, we also solved the same instances without introducing cut constraints using the non-convex feature of Gurobi 9.0.2. Although our intent was not to compare our solution method to the non-convex feature but to solve the problem faster, doing so revealed the efficiency of our technique. Table 2 illustrates the average gap and the solution time for the corresponding number of instances and the set sizes for both solution approaches. In this context, cutting-planes help converging to optimality sooner than the solver's non-convex feature. Apart from solution approach comparisons, this table also shows the capability of our model by indicating the gap and time values.

Table 2: Computational results.

$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{D}\rvert$	Instances	Average Gap		Average Time (s)	
			With Cut	Without Cut	With Cut	Without Cut
15	5	100	2.4×10^{-4}	2.9×10^{-7}	20.3	21.4
	10	50	3.0×10^{-4}	2.2×10^{-5}	20.5	21.2
	15	25	9.9×10^{-5}	6.5×10^{-6}	34.7	38.7
	20	10	2.1×10^{-4}	1.0×10^{-15}	42.7	66.1
20	5	100	2.7×10^{-4}	9.5×10^{-5}	23.8	24.0
	10	50	2.9×10^{-4}	1.2×10^{-6}	24.6	25.0
	15	25	5.0×10^{-5}	2.4×10^{-10}	46.9	109.6
	20	10	2.6×10^{-2}	6.2×10^{-2}	778.9	2,173.0
30	5	100	2.2×10^{-4}	2.4×10^{-5}	20.3	20.4
	10	50	1.7×10^{-4}	5.8×10^{-5}	25.1	28.1
	15	25	2.0×10^{-4}	9.1×10^{-8}	40.6	54.0
	20	10	4.9×10^{-5}	6.9×10^{-5}	180.5	643.8

As seen in this table, we consider 15, 20, and 30 ABSM candidate locations and 5, 10, 15, and 20 demand locations. For small problem-sizes (e.g., $|\mathcal{D}| = 5$), we generate more problem instances. The average gap and the average solution

time of these instances show that the solution method supported by Gurobi solver effectively solves the problem instances to a satisfactory solution gap percent and a short computational time.

5.2. Case Study

Sine we have access to Chicago metropolitan area population and geographical data (analyzing the region at a zonal-level) [2], we select this region as a test-bed. We randomly (from a uniform distribution) select $|\mathcal{D}|$ customer locations, which are centroids of zones, and estimate the overall demand for each zone given its population. In terms of ABSM network and the FC, we distribute ABSMs uniformly throughout the area and place the FC to the center of the network. Notice that the proposed methodology is not bound to these assumptions, and these are considered only to form a relatively realistic case study.

We share optimal routes for two cases as shown in Figure 1. In the first case (left), $|\mathcal{I}| = 22$ and $|\mathcal{D}| = 10$, and in the second case (right), $|\mathcal{I}| = 42$ and $|\mathcal{D}| = 25$. The first case was intentionally selected as a small-size to better visualize the optimal routing, and the second one is relatively large to reveal the applicability of our approach in a real-world problem. We set $\Delta = 5.0 \times 10^{-4}$ for both cases, obtained 2.3×10^{-4} (first case) and 1.1×10^{-5} (second case) optimality gap values, and received solutions in 21.3 (first case) and 453.4 (second case) seconds. In the figure, R denotes the maximum travel time without battery swapping only for the FC. Although the same range applies to ABSMs, we avoided plotting them as they complicate the presentation. Dashed lines with different colors indicate the routing to demand points.

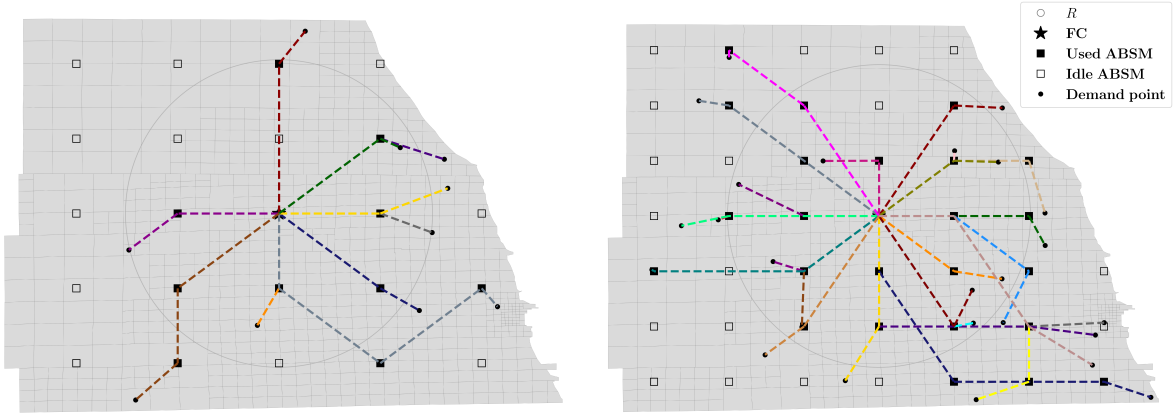

Fig. 1: Optimal routing in two cases.

6. Conclusion

In this study, we formulated a model and developed a solution method to solve a future DDRP problem. We employed queueing theory in an MINLP modeling framework and presented a cutting-plane-based exact solution. The objective in routing is to minimize the delivery lead time considering the bottleneck caused by battery-constrained travel ranges and the delivery delay due to a possible congestion at ABSMs.

The limitations of the study can be summarized as follows. This study only focuses on a single FC since considering multiple FCs will require assigning demand points to FCs. In practice, each demand point is served by different FCs over time due to many reasons, such as the real-time product availability of the order at FCs, cost of inbound transports to FCs, and so on. However, finding the long-term optimal demand-to-FC matching could be useful for practitioners while making delivery planning decisions. Another limitation stems from the use of a single drone type. For instance, Amazon showcases different types of delivery drone platforms with unique features, e.g., fixed wing and quadcopters. The speed and load carriage limits of these platforms considerably vary. Embedding this feature into a long-term planning framework could inform users on which type of platform to select under different circumstances.

Obstacle avoidance is another concern that needs to be tackled especially in urban deployment of the technology. Due to regulations, drones are not allowed to fly over certain areas. To address this issue, shortest travel times between points need to be calculated while avoiding the obstacles. The literature is full of studies solving this problem, and the concern can be easily addressed with parameter pre-processing without requiring change in the presented model formulation.

Future studies will first focus on deriving a more efficient algorithm to be able to solve much larger cases (in terms of demand location sizes) quickly. Then, the aforementioned limitations will be addressed.

Acknowledgement

The author thanks to Aymeric Rousseau and Joshua Auld of the Argonne National Laboratory for providing geographical data and to Asylon, Inc. for providing service time data.

References

[1] Asylon, Inc, 2020. An aerial infrastructure company. Available at https://www.flyasylon.com, accessed on Oct. 6, 2020.
[2] Auld, J., Hope, M., Ley, H., Sokolov, V., Xu, B., Zhang, K., 2016. POLARIS: Agent-based modeling framework development and implementation for integrated travel demand and network and operations simulations. Transportation Research Part C: Emerging Technologies 64, 101–116. doi:10.1016/j.trc.2015.07.017.
[3] Cokyasar, T., 2021. Optimization of battery swapping infrastructure for e-commerce drone delivery. Computer Communications 168, 146–154. doi:10.1016/j.comcom.2020.12.015.
[4] Cokyasar, T., Dong, W., Jin, M., 2019. Network optimization for hybrid last-mile delivery with trucks and drones, in: 2019 IISE Annual Conference & Expo Conference Proceedings, Institute of Industrial and Systems Engineers (IISE). pp. 1250–1255. ISBN: 978-0-9837624-8-5.
[5] Cokyasar, T., Dong, W., Jin, M., Verbas, I.O., 2021. Designing a drone delivery network with automated battery swapping machines. Computers & Operations Research 129, 105177–105187. doi:10.1016/j.cor.2020.105177.
[6] Curlander, J.C., Gilboa-Amir, A., Kisser, L.M., Koch, R.A., Welsh, R.D., 2017. Multi-level fulfillment center for unmanned aerial vehicles. US Patent 9,777,502.
[7] Gentry, N.K., Hsieh, R., Nguyen, L.K., 2016. Multi-use UAV docking station systems and methods. US Patent 9,387,928.
[8] Gurobi Optimization, LLC, 2020. Gurobi optimizer reference manual. Available at http://www.gurobi.com, accessed on Oct. 6, 2020.
[9] Hong, I., Kuby, M., Murray, A.T., 2018. A range-restricted recharging station coverage model for drone delivery service planning. Transportation Research Part C: Emerging Technologies 90, 198–212. doi:10.1016/j.trc.2018.02.017.
[10] Huang, H., Savkin, A.V., Huang, C., 2020a. A new parcel delivery system with drones and a public train. Journal of Intelligent & Robotic Systems 2020, 1–14. doi:10.1007/s10846-020-01223-y.
[11] Huang, H., Savkin, A.V., Huang, C., 2020b. Round trip routing for energy-efficient drone delivery based on a public transportation network. IEEE Transactions on Transportation Electrification 6, 1368–1376. doi:10.1109/TTE.2020.3011682.
[12] Kleinrock, L., 1975. Queueing systems. Volume I: Theory. John Wiley & Sons, New York, NY. ISBN: 978-0-471-49110-1.
[13] Murray, C.C., Chu, A.G., 2015. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. Transportation Research Part C: Emerging Technologies 54, 86–109. doi:10.1016/j.trc.2015.03.005.
[14] Murray, C.C., Raj, R., 2020. The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. Transportation Research Part C: Emerging Technologies 110, 368–398. doi:10.1016/j.trc.2019.11.003.
[15] Shavarani, S.M., Mosallaeipour, S., Golabi, M., İzbirak, G., 2019. A congested capacitated multi-level fuzzy facility location problem: An efficient drone delivery system. Computers & Operations Research 108, 57–68. doi:10.1016/j.cor.2019.04.001.