Approximation Algorithms for Drone Delivery Packing Problem

Saswata Jana
Indian Institute of Technology Guwahati
Guwahati – 781039, India
saswatajana@iitg.ac.in

Partha Sarathi Mandal
Indian Institute of Technology Guwahati
Guwahati – 781039, India
psm@iitg.ac.in

ABSTRACT
Recent advancements in unmanned aerial vehicles, also known as drones, have motivated logistics to use drones for multiple operations. Collaboration between drones and trucks in a last-mile delivery system has numerous benefits and reduces a number of challenges. In this paper, we introduce drone-delivery packing problem (DDP), where we have a set of deliveries and respective customers with their prescribed locations, delivery time intervals, associated cost for deliveries, and a set of drones with identical battery budgets. The objective of the DDP is to find an assignment for all deliveries to the drones by using the minimum number of drones subject to the battery budget and compatibility of the assignment of each drone. We prove that DDP is NP-Hard and formulate the integer linear programming (ILP) formulation for it. We proposed two greedy approximation algorithms for DDP. The first algorithm uses at most 2OPT + (Δ + 1) drones. The second algorithm uses at most 2OPT + ω drones, where OPT is the optimum solution for DDP, ω is the maximum clique size, and Δ is the maximum degree of the interval graph G constructed from the delivery time intervals.

CCS CONCEPTS
• Mathematics of computing → Combinatorial optimization;
• Computing methodologies → Optimization algorithms;
• Theory of computation → Scheduling algorithms.

KEYWORDS
Approximation Algorithm; Drone Delivery; Truck; Last-mile Delivery System; Time complexity

1 INTRODUCTION
The rapid demand for commercial deliveries motivates the e-commerce giants to deliver parcels more effectively to customers. Last-mile delivery [4], is the final step in this delivery journey, where the product needs to deliver to the customer’s doorstep from the distribution hub. This is the most expensive process and requires a lot of human interaction. But, advances in drone technologies make a miniature, and this delivery system brings more impact in today’s pandemic world. Big companies started their preparation to make productive parcel deliveries through drones [1]. In addition, using key vehicles (like trucks, cars, vans, etc.) along with drones with their constraints enhances the profit and diminishes the total delivery time interval. Further, drones have enormous applications in the field of defence and disaster response [11], agriculture [9], healthcare [15], etc.

Challenges: Delivery of packages by drones in association with a truck creates many challenges in our real-world scenario. For a given location of customers, we need to know the optimum route for the truck, the launching and rendezvous points of the drones, and the truck for making multiple deliveries. Also, the limited battery budget of the available drones in the market does not allow us to make a desirable number of deliveries. To guarantee all the deliveries for a given delivery time with a fixed number of drones, we need to either create rechargeable stations or a battery replacement policy for the drones, which makes the system complicated. In addition to this, we can’t use a drone for any set of deliveries altogether because of conflicts among delivery time intervals. At a time, a drone can deliver at most one package. All these complexities influence logistics to complete all deliveries by using a few identical drones so that the total delivery cost by the company is optimized.

Drone delivery scheduling problem: This problem considers last-mile delivery to customers using a set of drones carried by a truck moving in a prescribed route. For a given set of deliveries and their delivery time intervals, reward or profit for each delivery, and battery budget of the drones, the goal is to schedule a given fixed set of drones for the deliveries to make the total profit maximum. This problem was introduced by Sorbetti et al. [2] and proved that the problem is NP-Hard and proposed heuristic algorithms to solve the problems.

The aforementioned problem does not guarantee delivery of all packages because it uses a fixed set of drones to optimize total profit for delivery. In this paper, we introduce the Drone-Delivery Packing Problem (DDP), which aims to optimize the number of drones to deliver all deliveries. We propose two approximation algorithms to solve DDP.

Contributions. In this paper, our contributions are the following.
• In this paper, we introduce Drone Delivery Packing Problem (DDP) and prove that it is NP-Hard. We present an integer
linear programming formulation for it, which is only suitable for solving the problem optimally for small-sized instances.

- We propose an approximation algorithm for DDP with running time $O(n \log n + n_\varepsilon)$ and uses at most $2OPT + (\Lambda + 1)$ drones, where $n$ is the number of deliveries, $n_\varepsilon$ is the total number of edges and $\Lambda$ is the maximum degree of the interval graph $G$.

- We propose another approximation algorithm for DDP with running time $O(n \log n + n_\varepsilon)$ and uses at most $2OPT + \omega$ drones, where $n$ is the number of deliveries, $n_\varepsilon$ is the total number of edges and $\omega$ is the maximum clique size of the interval graph $G$.

**Related Work.** Since drones have certain mobility, using the truck makes more efficient deliveries. Various research has been done in this area of delivery with collaboration between drones and a truck.

This kind of delivery comes into account when Murray and Chu in [13] introduced flying sidekicks traveling salesman problem, a more extension of TSP, where customers need to visit (or deliver) either by the truck or by a drone starting from the depot. A drone begins its journey either from the depot or from any customer location, similarly, for the meeting occasion. Here authors want to minimize the total makespan to make all the deliveries. For this purpose, they proposed an optimal mixed integer linear programming (MILP) formulation and two effective heuristic solutions.

Crisan and Nechita [7] proposed another effective heuristic for flying sidekicks traveling salesman problem by using the solution for TSP. Murray and Raj [14] extended flying sidekicks traveling salesman problem for multiple drones. Here they proposed MILP formulation for the problem and then a heuristic solution with numerical testing. Daknama and Kraus [8] take in hand of mobility of drones. There is a rechargeable area on the truck’s roof where the drones can charge after completing one delivery and go for the next delivery. Here authors proposed a heuristic algorithm for scheduling of truck and drones. Delivery by drones only in the attention in [3], where Boysen et al. objective to find the launch and meet point for delivery with the truck so that total makespan for completing all the deliveries minimized. The assumption for this problem was the knowledge of the truck’s route but without any battery constraint of the drones. Mathew et al. [12] proposed a heterogeneous delivery problem, where two co-operative vehicles (truck and micro-aerial vehicles (MAV)) are used for performing all the deliveries. Whereas drones can fly from and meet with the truck at the prescribed warehouses. Their goal is to find the optimal route with respect to the cost. For this, they proposed two heuristic solutions and the hardness of the problem.

Very recently, Sorbelli et al. [2] proposed a Multiple Drone-Delivery Scheduling Problem (MDSP), where a truck and multiple drones cooperate among them-self for package delivery in a last-mile. The paper gave NP-hardness proof of the problem, ILP formulation, and designed a heuristic algorithm for the single drone case and two heuristic algorithms for the multiple drones case.

The problem we discuss here is a more generalized version of the bin-packing problem. Coffman et al. in [6] presents several variants of bin-packing problem with their approximation algorithm. Stacho in [16] described various colouring of the chordal graph along with their complexities.

**Roadmap.** We discuss the model and preliminaries with problem definition in Section 2. We present the hardness of the Drone-Delivery Packing Problem and ILP formulation of it in Section 3. We propose two approximation algorithms in Section 4. Finally, we conclude in Section 5.

## 2 MODEL AND PRELIMINARIES

**Model:** Let $\mathcal{N} = \{1, 2, \ldots, n\}$ be the set of deliveries to be delivered to the respective customers with their prescribed location at $\delta_j$ for each $j \in \mathcal{N}$. A delivery company wants to deliver packages to the corresponding customers by the drones having identical battery capacity or budget $B$. Let $c_j$ $(0 < c_j \leq B)$ be the energy cost for a drone to complete the delivery $j \in \mathcal{N}$. Initially, all the drones are at the company’s warehouse (depot). Now, a truck will leave the depot with all the drones in its pre-decided path. For making a delivery $j$ at position $\delta_j$, a drone will fly from the truck at a specific launching location $(\delta^L_j)$ and after completing the delivery at $\delta_j$ it meets with the truck again at a specific rendezvous location $(\delta^R_j)$. After completing all the deliveries, the truck with the drones returns to the depot.

Let at time $t_0$ the truck starts its journey with all the drones. For each delivery $j$, at the time $t^L_j$ and $t^R_j$ one drone comes at the position $\delta^L_j$ and $\delta^R_j$, respectively. Then, $I_j = [t^L_j, t^R_j]$ be the delivery time interval for the delivery $j \in \mathcal{N}$. Let $I = \{I_1, I_2, \ldots, I_n\}$ be the delivery time interval set for the set of deliveries $\mathcal{N}$. The truck moves in one direction. So, if $A$ and $B$ are any two points on the truck’s pre-decided path, where $B$ is the later point of $A$, then $t_A < t_B$, where $t_A$ and $t_B$ are the time when the truck arrives at the position $A$ and $B$, respectively.

Any drone can be assigned for multiple deliveries $S (S \subseteq I)$ constraints to the battery budget $B$ of the drone and the compatibility of the delivery time intervals. Any two delivery time intervals $I_j$ and $I_k$ are said to be compatible or conflict-free if $I_j \cap I_k = \emptyset$, otherwise they are in conflict. Any set of delivery time intervals $S \subseteq I$ is said to be compatible if all pairs of intervals in it are compatible. A compatible set $S \subseteq I$ is said to be feasible if $\sum_{j \in S} c_j \leq B$. A feasible set $S \subseteq I$ is assigned for the drone $i$ then, we call $S$ as an assignment of the drone $i$. We are saying a drone is in used or opened if at least one delivery assigned to this drone.

**Graph.** For the given set of delivery time intervals $I$, we can construct an interval graph $G = (V, E)$, where the vertices represent the intervals and two vertices are adjacent if the corresponding two intervals conflict. $A$ is denoted as the maximum degree of $G$, $|V| = |I| = n$ and $|E| = n_\varepsilon$. Now, $G = (V, E)$ being an interval graph, it is perfect [17]. Therefore, $\chi(G) = \omega(G)$, where $\chi(G)$ is the chromatic number and $\omega(G)$ is the maximum clique size of $G$. We use $\omega$ instead of $\omega(G)$ for simplicity. So, the vertices of $G$ can be optimally coloured by using $\omega$ many colours in polynomial time (linear in terms of the number of vertices and edges) [16] such that no two adjacent vertices get the same colour. Since every interval in $I$ is represented by a unique vertex in $G$, each interval can be coloured linearly from the colour of $G$. Further, we can partition $I$ into $\omega$
many conflict-free (i.e., compatible) sets corresponding to each colour. Let \( J_1, J_2, \cdots, J_ω \) be the partitions. Therefore, \( \cup_{k=1}^ω J_k = I \).

**Example.** Figure 1 shows an example of a drone-delivery model with eight delivery locations and two drones. The solid lines in the figure represent the paths of the truck, while the dotted lines represent the paths of the drone. Figure 2 illustrates intervals corresponding to the delivery intervals of Figure 1. The interval graph \( G = (V, E) \) corresponding to the delivery intervals of Figure 2 is shown in the Figure 3.

![Figure 1: A drone-delivery model with paths specification.](image1)

**Figure 1: A drone-delivery model with paths specification.**

\( I = \{I_1, I_2, \cdots, I_8\} \) where \( I_j = [I_j^L, I_j^R] \) for \( 1 \leq j \leq 8 \).

![Figure 2: Delivery intervals \( I = \{I_1, I_2, \cdots, I_8\} \) where \( I_j = [I_j^L, I_j^R] \) for \( 1 \leq j \leq 8 \).](image2)

**Figure 2: Delivery intervals \( I = \{I_1, I_2, \cdots, I_8\} \) where \( I_j = [I_j^L, I_j^R] \) for \( 1 \leq j \leq 8 \).**

![Figure 3: Interval Graph, \( G = (V, E) \).](image3)

**Figure 3: Interval Graph, \( G = (V, E) \).**

**Drone-Delivery Packing Problem (DDP).** DDP is formally defined as follows.

**Definition 2.1 (DDP).** Given a set of delivery time intervals \( I = \{I_1, I_2, \cdots, I_8\} \) corresponding to the set of deliveries \( N \) associated with cost \( c_j \) for each \( j \in N \), the objective for DDP is to use the minimum number of drones such that each delivery is completed by exactly one drone and each assignment of opened drones is feasible.

In another way, find a smallest cardinality set of drones \( M^* = \{1, 2, \cdots, m^*\} \) along with a family of assignments \( S^* = \{S_1^*, S_2^*, \cdots, S_m^*\} \), where \( S_j^* \subseteq I \) is a feasible assignment for the drone \( i \in M^* \) such that each delivery is associated with a unique element (assignment) in \( S^* \) and \( S_j^* \cap S_m^* = \phi; \forall 1 \leq l \neq m \leq m^* \).

## 3 PROBLEM HARDNESS

Here we establish the hardness of the DDP via the bin packing problem [5].

**Theorem 3.1.** DDP is an NP-hard problem.

**Proof.** We prove it by the reduction from the bin-packing problem (BP).

The goal of the bin-packing problem is to pack given a set of items associated with some sizes into a set of bins with identical capacity such that the number of bins used for the packing is minimum.

Let \( I_{BP} = (N_{BP}, s, b) \) be an arbitrary instance of BP, where \( N_{BP} = \{1, 2, \cdots, n\} \) is the set of items, \( b \) is the capacity of each identical bin and \( s: N_{BP} \rightarrow (0, b] \) is the size function of the items with \( s(j) = s_j, \forall j \in N_{BP} \).

The above instance \( (I_{BP}) \) for BP can be transformed into an instance of DDP as follows. Let \( I = (N, I, c, b) \), where \( N = N_{BP} \) is the set of deliveries, \( I = \{I_1, I_2, \cdots, I_n\} \) is the set of delivery time intervals with \( I_j = [2j, 2j+1] \) being the delivery time interval of the delivery \( j (1 \leq j \leq n) \), \( B = b \) is the battery budget of identical drones and \( c: N \rightarrow (0, B] \) is the cost function of the deliveries with \( c(j) = c_j = s_j \), \forall j \in N. I \) is an instance of DDP and the reduction from \( I_{BP} \) is polynomial.

All the delivery time intervals in \( I \) are pairwise compatible. So, if all the items in \( N_{BP} \) can be packed using \( m \) bins, then all the deliveries in \( N \) can be scheduled using \( m \) drones and vice-versa. Thus, BP \( \leq_P \) DDP. Hence, being BP an NP-hard problem [10], DDP is so.

### 3.1 ILP Formulation

For solving DDP optimally, we can formulate the problem via integer linear programming (ILP) as follows. According to the assumption, the associated cost \( c_j \) for each delivery \( j \in N \) does not exceed the battery budget of the drones. So, \( n = |N| \) is an upper bound for the optimal solution of the DDP. Let \( M = \{1, 2, \cdots, n\} \) be the set of drones available in the company’s warehouse. Let,

\[
\begin{align*}
y_i &= 1, \text{ if drone } i \in M \text{ is used} \quad \Rightarrow 0, \text{ otherwise.} \\
x_{ij} &= 1, \text{ if delivery } j \in N \text{ completed by the drone } i \in M \quad \Rightarrow 0, \text{ otherwise.} \\
\min \sum_{i \in M} y_i \\
\end{align*}
\]

subject to \( \sum_{j \in N} c_{jx_{ij}} \leq B y_i, \quad \forall i \in M \).
Algorithm 1 takes a simple greedy approach like First-Fit bin packing algorithm [5]. At first, the algorithm sorts the deliveries according to their launching time, takes the delivery with the shortest launch time, and assigns it to any of the drones. The assigned drone is termed as used or opened drone. Then it takes the delivery one by one as per the sorted order and tries to assign them to the used drones. A new drone is introduced if any delivery does not fit, i.e., not feasible with any of the used drones.

To get a better time complexity of the above greedy approach, we are using a balanced binary search tree with each node corresponding to a used drone. The key of that node is the remaining battery capacity of the used drone. Furthermore, for each node, we store an additional data corresponding to the rendezvous time of the last assigned delivery to the associated drone. All other attributes of the node are specified below. With the help of this tree, we can efficiently find the index of the used drone in which we can assign a particular delivery (in addition to the existing assignment) or get the confirmation to introduce a new drone. The pseudocode of this greedy approach with the tree depicts in Algorithm 1.

**Variable Specification:** 
- $T$: A balanced binary search tree $T$ is represented by root node (initially root = NULL). node/ NODE: Tree nodes with five attributes. For node, we use node/index: index of the corresponding drone; node/key: remaining battery capacity of the drone; node/data: rendezvous time of the last assigned delivery to the drone; node/left: left child of the node in the tree; node/right: right child of the node in the tree. Similarly, we can consider the attributes: NODE/index, NODE/key, NODE/data, NODE/left, NODE/right for NODE. The tree $T$ maintains it’s balance and search property according to the node’s key attribute.
- Global variable: $d_j = N(j) + 1$: Maximum number of checks need to be done in the tree for the delivery $j$ to maintain compatibility property of an assignment, whereas $N(j)$ is the number of conflicts for the delivery $j$, $\forall j: 1 \leq j \leq n$.

**Description of Algorithm 1:** Initially, no drone is opened or used (i.e., $m = 0$), where $m$ represents the number of used drones, and the tree $T$ is empty, represented by root = NULL. As one node represents one drone, in the remaining part of this paper, we will use ‘assign to a drone’ and ‘assign to a node’ as a synonym. While assigning a delivery to a node means assigning that delivery to the drone corresponds to that node. A node is called feasible for the delivery $j$, if the node’s existing assignment with $l_j$ is a feasible assignment. For assigning a delivery $j$ according to the non-decreasing launching time, the algorithm calls the function $\text{Find}(\text{root}, j)$ to find the node where we can assign the delivery. If $\text{Find}$ returns a non-null pointer of a node (node), then we assign the delivery $j$ to node/index and call $\text{Update}()$. In this function we decrease the node/key by (node/key – $c_j$) and update node/data by $t^B_{ij}$. Decreasing the key of a node may violate the search property of the balanced binary search tree. Then, we delete the node first and then insert a new node with the same attributes of the deleted node.

Otherwise (Find returns a null pointer), a new node (node) is introduced with index $(m + 1)$ and assigns the delivery $j$. Then call $\text{Insert}()$ for inserting a new node in the balanced binary search tree with node/key as $(B – c_j)$ and node/data as $t^B_{ij}$.

**Algorithm 1: GreedyAlgoForDDP($i$, $c$, $B$)**

| Line | Description |
|------|-------------|
| 1    | Initialize: $m = 0$; $S_l = \phi, \forall i : 1 \leq i \leq n$; root = NULL |
| 2    | Sort the intervals according to their launching time. (without loss of generality let $t^B_{i1} \leq t^B_{i2} \leq \cdots \leq t^B_{in}$) |
| 3    | for $j \leftarrow 1, n$ do |
| 4    | node = $\text{Find}($root, $j$) |
| 5    | if node $\neq$ NULL then |
| 6    | $i \leftarrow$ node/index; $S_l \leftarrow S_l \cup \{l_j\}$ |
| 7    | DATA $\leftarrow t^B_{ij}$ |
| 8    | $\text{Update}($root, $node$, node/key $\cdot c_j$, DATA) |
| 9    | else |
| 10   | $m \leftarrow m + 1$ |
| 11   | $S_m \leftarrow \{l_j\}$ |
| 12   | $\text{Insert}($root, $m$, $B – c_j$, $t^B_{ij}$) |
| 13   | return $m$, along with the assignments $\{S_1, S_2, \cdots, S_m\}$ |

**Description of Algorithm 2:** $\text{Find}()$ takes node and $j$ as the inputs. It finds the feasible node via reverse in-order traversal in the balanced binary search tree where the delivery $j$ can be assigned. It returns the pointer of a node if the node is found in the tree rooted at node is feasible for the delivery $j$ or NULL if no feasible node is found.

If the tree is empty or the node is NULL, the algorithm returns NULL. Otherwise, the algorithm executes in three parts. It may return a non-null or null pointer at any of these parts.

- **$\text{Find}($node.right, $j$)** At first, the algorithm finds the feasible node in the right subtree of node (line 4) by calling $\text{Find}($node.right, $j$). If it returns a non-null pointer, the algorithm returns that pointer (line 5 – 6). Else, the algorithm proceeds for the next part to check whether the node itself is feasible for the delivery $j$ by calling $\text{Check}($node, $j$).
If $node.key < c_j$, no further checking is required, as all the other nodes after the node (in reverse inorder) have lesser or equal to remaining capacity than $c_j$. So, all these nodes are not feasible for the delivery $j$. For this case, Check returns FALSE and set $d_j = 0$, as no further checking is required for the delivery $j$. If $node.data \geq t^f_j$, then checking to be done at the predecessor of the node and decrease the maximum checking number ($d_j$) for the delivery $j$ by one (line 7) and the algorithm returns FALSE.

Hence, algorithm 3 confirms that the assignment for each node is feasible, and so is the assignment of each used drone.

Algorithm 3: Check(node, j)

```
1 if node.key \geq c_j and node.data < t^f_j then
  return TRUE
2 else if node.key < c_j then
  d_j \leftarrow 0
3 else
  return FALSE
```

Lemma 4.1. If $m$ is the number of used (opened) drones returned by the algorithm 1, then at least $(m - \Delta - 1)$ drones have used energy cost at least $\frac{B}{\Delta}$, where $\Delta$ is the maximum degree of the interval graph $G$ constructed from delivery intervals.

Proof. We prove the lemma by contradiction. Let at most $(m - \Delta - 2)$ used drones have used energy cost at least $\frac{B}{\Delta}$. Then, at least $(\Delta + 2)$ drones have used energy cost $< \frac{B}{\Delta}$, without loss of generality let $\{1, 2, \cdots, \Delta + 2\}$ be those drones. Now, consider the time ($t$) when the drone $(\Delta + 2)$ was introduced by the algorithm. Let for the delivery $j$ it was introduced. Then either $S_i^j \cap I_j \neq \emptyset$ or $W^f_i + c_j > B (1 \leq i \leq (\Delta + 1))$ holds, where $S_i^j$ is the existing assignment and $W^f_i$ is the total cost for the existing assigned deliveries of the drone $i$ at time $t$ of the algorithm. $I_j$ can conflict with at most $\Delta$ many $S_i^j$s, as $\Delta$ is the maximum degree of $G$, i.e., $\Delta$ is the maximum conflict number of any interval in $I$. Then, we can find an $i' (1 \leq i' \leq \Delta + 1)$ for which $W^f_i + c_j > B$ holds. But, at the end of the algorithm (say at time $t_e$), $W^{t_e}_i < \frac{B}{\Delta}$ and $W^{t_e}_f < \frac{B}{\Delta}$, implies $c_j < \frac{B}{\Delta}$, implies $W^{t_e}_i + c_j < B$, implies the drone $i'$ is feasible the delivery $j$, which is a contradiction. Hence, the statement of the lemma follows. \hfill \sq

Lemma 4.2. The time complexity for algorithm 1 is $O(n \log n + n_e)$, where $n$ is the number of deliveries and $n_e$ is the number of edges in $G$.

Proof. At first, the algorithm sorts the intervals according to their launch time, takes $O(n \log n)$ time.

Consider the time of execution when the algorithm wants to assign the delivery $j$. At this point, the number of drones opened is at most $(j - 1)$, and so is the number of nodes in the tree. The tree is being a balanced binary search tree, at this time height of the tree is $O(\log j)$. For finding the appropriate node in the tree, which is feasible for the delivery $j$, the algorithm checks at most
Algorithm 1 is an approximation algorithm for DDP, uses at most 2OPT + (Δ + 1) drones, where \( n \) is the number of deliveries, OPT is the optimum number of drones required for DDP, and \( Δ \) is the maximum degree of the interval graph \( G \).

Proof. Initially, the algorithm sorts the deliveries based on their non-decreasing launch time. Without loss of generality, let \( t_{i1}^1 \leq t_{i2}^1 \leq \cdots \leq t_{in}^1 \). Then, it assigns all deliveries in accordance with the previously sorted order. First, the algorithm takes the delivery with the shortest launching time and assigns it to a drone (indexed by \( i \)). We call this drone as a used drone. Since the cost for each delivery is \( \leq B \), this is a feasible assignment. After that, for assigning the delivery \( i_j \) (\( 2 \leq j \leq n \)) as per their sorted order, the algorithm checks whether there exists a drone \( i \) among all the used drones (say \( m \)), which is feasible for this delivery. If yes, assign the delivery to the drone \( i \). Otherwise, a new drone is introduced with index \( (m + 1) \) and assigned the delivery \( i_j \). The feasibility of each existing assignment is always upheld for each such addition of the delivery. Thus, all the returned assignments are feasible at the end of the algorithm 1.

The polynomial running time of the algorithm follows from Lemma 4.2.

If \( m \) is the number of drones (used drones) returned by the algorithm, then from the lemma 4.1 following holds.

\[
(m - Δ - 1) \cdot \frac{B}{2} \leq \sum_{j=1}^{n} c_j \leq OPT \cdot B
\]

\[
\Rightarrow m \leq 2 \cdot OPT + (Δ + 1)
\]

Hence the proof. \( \square \)

4.2 Approximation Algorithm for DDP using Colouring

In this section, we demonstrate another approximation algorithm (Algorithm 6) using colouring to solve DDP. The algorithm in general gives a better solution than the previous one.

From the given interval set \( I \), we can construct an interval graph \( G \) as described in section 2. Interval graph being a perfect graph [17], vertices of \( G \) can be coloured by \( ω \) many colours, where \( ω \) is the maximum clique size of \( G \). From the \( ω \)-colouring of \( G \), we can partition the set \( I \) into many sets, each corresponding to the same coloured vertices of \( G \). Let \( \{J_1, J_2, \cdots, J_\omega\} \) be the partition set, where \( J_k \subseteq I \) is a compatible set associated to colour \( k \) (\( 1 \leq k \leq \omega \)).

For each \( k \) (\( 1 \leq k \leq \omega \)), we can find the number of drones needed to schedule all the intervals in \( J_k \) and the corresponding schedules using algorithm 4, a modified version of algorithm 1. Similar to the algorithm 1, for each interval set \( J_k \), we construct a balanced binary search tree \( T_k \) (\( 1 \leq k \leq \omega \)).

The pseudocode of the modified version of algorithm 1 is depicted in algorithm 4. Following changes are made on algorithm 1 to modify it to algorithm 4. As \( T_k \) is compatible, the data from each tree node can be omitted. The other attributes (node key, node index, node.left, node.right) of a node in each tree remains same. For a fixed interval set \( J_k \) (\( 1 \leq k \leq ω \)), the index of \( i-th \) used drone for the interval set \( J_k \) is denoted by \( i_k \) and the corresponding assignment is denoted by \( S_k \). For assigning a delivery \( i_j \) in \( J_k \), algorithm 4 calls FINDMODIFIED(root, \( e_j \)). If FINDMODIFIED(root, \( e_j \)) returns NULL, a new drone is introduced with index \( (m + 1)_k \) (initially, \( m = 0 \)) and assigns the delivery \( i_j \) to the drone. Then the algorithm calls INSERTMODIFIED() for inserting a new drone with index \( (m + 1)_k \) and key as \( B - e_j \). If FINDMODIFIED(root, \( e_j \)) returns a non-null pointer of a node then the algorithm assigns the delivery \( i_j \) at the node and calls UPDATEMODIFIED() for decreasing the node key by \( e_j \). Whereas FINDMODIFIED() finds the node with maximum key in the tree. If the node with maximum key is not feasible for the delivery \( i_j \), then no other node in the tree will be feasible for \( i_j \). So, for this case, FINDMODIFIED() returns NULL. Otherwise, FINDMODIFIED() returns the pointer of the node. Note that, if algorithm 4 returns \( m_k \), then the number of used drones for the interval set \( J_k \) is \( m_k \).

Let \( m_k \) be the number of drones returned by the algorithm 4 for the delivery interval set \( J_k \). Then, the following Lemma 4.4 is true.

**Lemma 4.4.** \( W(J_k) > \left( \frac{m_k - 1}{2} \right) \cdot B \), where \( m_k \) is the number of drones returned by algorithm 4 for the interval set \( J_k \leq 1 \) and \( W(J_k) = \sum_{i \in J_k} (W(S_1) + W(S_2)) \).

Proof. Let \( m_k \) drones are denoted by \( \{1_k, 2_k, \cdots, m_k\} \) and \( S_i \) = set of intervals in \( J_k \) which are assigned to the drone \( i \) by the algorithm 4 \( 1_k \leq i \leq m_k \). Then, \( W(S_{2i-1}) + W(S_{2i}) > B \) for \( (1 \leq i \leq \frac{m_k}{2} - 1) \), otherwise all the intervals in \( S_{2i-1} \) and \( S_{2i} \) can be assigned to a single drone. Thus,

\[
\sum_{i=1}^{\frac{m_k}{2}} (W(S_{2i-1}) + W(S_{2i})) > \left( \frac{m_k}{2} \right) \cdot B
\]

\[
\Rightarrow W(J_k) > \left( \frac{m_k}{2} - 1 \right) \cdot B
\]

\[
\Rightarrow W(J_k) > \left( \frac{m_k - 1}{2} \right) \cdot B
\]

**Lemma 4.5.** The time complexity for Algorithm 6 is \( O(n \log n + n_e) \), where \( n \) is the number of deliveries and \( n_e \) is the number of edges in \( G \).

Proof. We can construct an interval graph \( G \) (line 2) from the given interval set \( I \) in \( O(n + n_e) \) time. Then, find maximum clique size \( ω \) (line 3) and the colour all the vertices of \( G \) (line 4) with these \( ω \) colours. This can be done in \( O(n + n_e) \) time [16]. Finding all the
Approximation Algorithms for Drone Delivery Packing Problem

Algorithm 4: GreedyAlgModified(J_k, c, B)

1. Initialize: m = 0; S_{i_k} = φ, ∀ i : 1 ≤ i ≤ n; root = NULL
2. for each interval I_j in J_k do
   3. node = FINDModified(root, c_j)
   4. if node ≠ NULL then
      5. i_k ← node.index; S_{i_k} ← S_{i_k} ∪ {I_j}
      6. UPDATEModified(root, node, node.key + c_j)
   7. else
      8. m ← m + 1
      9. S_{m_k} ← {I_j}
   10. INSERTModified(root, m_k, B - c_j)
3. return m_k, with their assignments S_{i_k}

Algorithm 5: FINDModified(node, c_j)

1. if node = NULL then
   2. return NULL
3. else
   4. if node.right = NULL and node.key ≥ c_j then
      5. return node
   5. else if node.right = NULL and node.key < c_j then
      6. return NULL
   7. else
      8. return FINDModified(node.right, c_j)

Algorithm 6: ApproxAlgWithColouring

1. Input: Set of delivery intervals I = {I_1, I_2, · · · , I_n}; cost c_j for each I_j ∈ I; drone battery budget B;
2. Construct an interval graph G from the delivery time interval set I
3. Find maximum clique size (ω) of G.
4. Colour all the vertices of G with the colors {1, 2, · · · , ω} such that no two adjacent vertices get the same colour.
5. Find J_k = Set of intervals in G whose corresponding vertices in G are coloured by the colour k (1 ≤ k ≤ ω).
6. For each J_k (1 ≤ k ≤ ω), find number of drones, say m_k and corresponding assignments, say S_k by using the algorithm 4.
7. Return ω\sum_{k=1}^{ω} m_k along with their corresponding assignments.

J_k (1 ≤ k ≤ ω) (line 5) takes O(n) time.
Algorithm 6 uses algorithm 4 for finding the number of drones and corresponding assignments for each delivery interval J_k (1 ≤ k ≤ ω) (line 6). For each interval set J_k (1 ≤ k ≤ ω) we create a balanced binary search tree (T_k) similar to the previous section. For assigning a delivery I_j in J_k, algorithm 4 call FINDModified(root, c_j). This returns a pointer in O(h_k^3) time, where h_k is the height of the tree T_k before the assignment of delivery I_j. If FINDModified(root, C_j) returns NULL, the algorithm calls INSERT(). Otherwise, the algorithm calls UPDATE(). For any of the case, algorithm 4 needs O(h_k^3) time for assigning delivery I_j in J_k. Thus, for the interval set J_k algorithm 4 runs in \sum_{k=1}^{ω} O(h_k^3) ≤ O(n_k log n_k) time, where n_k is the number of deliveries in J_k. So, total running time for line 6 of algorithm 6 is \sum_{k=1}^{ω} O(n_k log n_k) = O(n log n).

Hence, overall running time for Algorithm 6 is O(n log n + n_c).

Theorem 4.6. Algorithm 6 is an approximation algorithm for DDP, which uses at most (2OPT + ω) drones, where ω is the maximum clique size of G and OPT is the optimum number of drones required for DDP.

Proof. Algorithm 6 uses algorithm 4 as it’s subroutine for each delivery interval set J_k (1 ≤ k ≤ ω). Algorithm 4 is a modified version of algorithm 1, and from theorem 4.3, we know that algorithm 1 always gives a feasible solution. So, {J_1, J_2, · · · , J_ω} being partition of the given delivery interval set I, algorithm 6 gives feasible assignments for each delivery in I.

The polynomial running time of the algorithm follows from lemma 4.5.
Let \mathcal{W}(I) be the total cost of all the deliveries in I and \mathcal{W}(J_k) be the total cost of all the deliveries in J_k, \forall k : 1 ≤ k ≤ ω.

Then, \sum_{k=1}^{ω} \mathcal{W}(J_k) = \mathcal{W}(I) ≤ OPT.B.

Let m_k be the number of drones returned by the algorithm 4 for the interval set J_k, \forall k : 1 ≤ k ≤ ω. Thus, from the lemma 4.4 following holds.

\sum_{i=1}^{ω} \left( \frac{m_k - 1}{2} \right) · B < \sum_{k=1}^{ω} \mathcal{W}(J_k) \quad (13)
\sum_{i=1}^{ω} \left( \frac{m_k - 1}{2} \right) · B < OPT.B \quad (14)
\Rightarrow \sum_{i=1}^{ω} m_k < 2OPT + ω. \quad (15)

Hence the proof.

5 CONCLUSION

In this paper, we studied the drone-delivery packing problem (DDP). We propose two approximation algorithms with identical running time O(n log n + n_c), where n is the number of deliveries and n_c is the number of edges in the interval graph G, constructed from the delivery time intervals. The first algorithm (Algorithm 1) uses 2OPT + (Δ + 1) drones, and the second algorithm (Algorithm 6) uses 2OPT + ω drones, where ω is the maximum clique size of G, Δ is the maximum degree of G and OPT is the optimum number of drones required for the DDP. In general, ω ≤ (Δ + 1), so, the second algorithm gives a better approximation than the first. However, the second algorithm gives 3-factor approximation as we need at least ω many drones for scheduling all the deliveries, i.e., ω ≤ OPT.

Finding better constant factor approximation algorithms and asymptotic polynomial time approximation schemes (PTAS) for DDP will be considered for future research. Furthermore, if the drone has a charging area inside the truck, determining the delivery schedule using the fewest possible drones subject to a finite amount of charging time is of great interest.
REFERENCES

[1] Amazon. [n.d.]. Amazon customers in Lockeford, California, will be among the first to receive Prime Air drone deliveries in the U.S. https://www.aboutamazon.com/news/transportation/amazon-prime-air-prepares-for-drone-deliveries

[2] Francesco Betti Sorbelli, Federico Corò, Saajal K. Das, Lorenzo Palazzetti, and Cristina M. Pinotti. 2022. Greedy Algorithms for Scheduling Package Delivery with Multiple Drones. In 23rd International Conference on Distributed Computing and Networking (Delhi, AA, India) (ICDCN 2022) Association for Computing Machinery. New York, NY, USA, 31–39. https://doi.org/10.1145/3491003.3491028

[3] Nils Boysen, Dirk Briskorn, Stefan Fedtke, and Stefan Schwerdflüger. 2018. Drone delivery from trucks: Drone scheduling for given truck routes. Networks 72 (2018), 506 – 527.

[4] Nils Boysen, Stefan Fedtke, and Stefan Schwerdflüger. 2021. Last-mile delivery concepts: a survey from an operational research perspective. OR Spectrum 43 (03 2021), 1–58. https://doi.org/10.1007/s00291-020-00607-8

[5] E. G. Coffman, M. R. Garey, and D. S. Johnson. 1984. Approximation Algorithms for Bin-Packing — An Updated Survey. Springer Vienna, Vienna, 49–106. https://doi.org/10.1007/978-3-7908-4338-4_3

[6] Edward G. Coffman Jr., János Csirik, Gábor Galambos, Silvano Martello, and Daniele Vigo. 2013. Bin Packing: Approximation Algorithms: Survey and Classification. Springer New York, New York, NY, 455–531. https://doi.org/10.1007/978-1-4419-7997-1_35

[7] Gloria Cerasela Crișan and Elena Nechita. 2019. On a cooperative truck-and-drone delivery system. Procedia Computer Science 159 (2019), 38–47. https://doi.org/10.1016/j.procs.2019.09.158 Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES2019.

[8] Rami Dahan and Elisabeth Kraus. 2017. Vehicle Routing with Drones. ArXiv abs/1705.06431 (2017).

[9] Gopal Dutta and Purba Goswami. 2020. Application of drone in agriculture: A review. International Journal of Chemical Studies 8 (10 2020), 181–187. https://doi.org/10.22271/chemi.2020.v8.i10d.10529

[10] Michael R. Garey and David S. Johnson. 1979. Computers and intractability. W. H. Freeman and Co., San Francisco, Calif. x+338 pages. A guide to the theory of NP-completeness.

[11] Fiore Kardasz and Jacek Doskos. 2016. Drones and Possibilities of Their Using. Journal of Civil & Environmental Engineering 6 (01 2016). https://doi.org/10.4172/2165-784X.1000233

[12] Neil Mathew, Stephen L. Smith, and Steven L. Waslander. 2015. Planning Paths for Package Delivery in Heterogeneous Multirobot Teams. IEEE Transactions on Automation Science and Engineering 12, 4 (2015), 1298–1308. https://doi.org/10.1109/TASE.2015.2461213

[13] Chase C. Murray and Amanda G. Chu. 2015. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. Transportation Research Part C: Emerging Technologies 54 (2015), 86–109. https://doi.org/10.1016/j.trc.2015.03.005

[14] Chase C. Murray and Ritwik Raj. 2020. The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. Transportation Research Part C: Emerging Technologies 110 (2020), 368–398. https://doi.org/10.1016/j.trc.2019.11.003

[15] Hyung Jin Park, Reza Mirjalili, Murray J. Côté, and Gino J. Lim. 2022. Scheduling Diagnostic Testing Kit Deliveries with the Mothership and Drone Routing Problem. J. Intell. Robotics Syst. 105, 2 (jun 2022), 19 pages. https://doi.org/10.1007/s10846-022-01632-1

[16] Juraj Stacho. 2008. Complexity of Generalized Colourings of Chordal Graphs. Ph.D. Dissertation. CAN. AADNRA6626.

[17] D.B. West. 1996. Introduction to Graph Theory. Prentice Hall, Chapter 5, 194–196.