# Package Delivery Using Drones with Restricted Movement Areas

**Thomas Erlebach**
Department of Computer Science, Durham University, UK

**Kelin Luo**
Institute of Computer Science, Universität Bonn, Germany

**Frits C.R. Spieksma**
Department of Mathematics and Computer Science,
Eindhoven University of Technology, The Netherlands

## Abstract

For the problem of delivering a package from a source node to a destination node in a graph using a set of drones, we study the setting where the movements of each drone are restricted to a certain subgraph of the given graph. We consider the objectives of minimizing the delivery time (problem DDT) and of minimizing the total energy consumption (problem DDC). For general graphs, we show a strong inapproximability result and a matching approximation algorithm for DDT as well as *NP*-hardness and a 2-approximation algorithm for DDC. For the special case of a path, we show that DDT is *NP*-hard if the drones have different speeds. For trees, we give optimal algorithms under the assumption that all drones have the same speed or the same energy consumption rate. The results for trees extend to arbitrary graphs if the subgraph of each drone is isometric.

**2012 ACM Subject Classification** Theory of computation → Design and analysis of algorithms

**Keywords and phrases** Mobile agents, approximation algorithm, inapproximability

**Digital Object Identifier** 10.4230/LIPIcs.ISAAC.2022.49

**Related Version** *Full Version*: https://arxiv.org/abs/2209.12314

**Funding** *Thomas Erlebach*: Supported by EPSRC grants EP/S033483/2 and EP/T01461X/1.
*Kelin Luo*: Supported by the National Natural Science Foundation of China, Grant No. 72071157.
*Frits C.R. Spieksma*: Supported by NWO Gravitation Project NETWORKS, Grant Number 024.002.003.

## 1 Introduction

Problem settings where multiple drones collaborate to deliver a package from a source location to a target location have received significant attention recently. One motivation for the study of such problems comes from companies considering the possibility of delivering parcels to consumers via drones, e.g., Amazon Prime Air [1]. In previous work in this area [8, 9, 3, 4, 2, 6, 7], the drones are typically modeled as agents that move along the edges of a graph, and the package has to be transported from a source node to a target node in that graph. Optimization objectives that have been considered include minimizing the delivery time, minimizing the energy consumption by the agents, or a combination of the two. A common assumption has been that every agent can travel freely throughout the whole graph [3, 4, 6], possibly with a restriction of each agent's travel distance due to energy constraints [8, 9, 2, 7]. In this paper, we study for the first time a variation of the problem in which each agent is only allowed to travel in a certain subgraph of the given graph.

We remark that the previously considered problem in which each agent has an energy budget that constrains its total distance traveled [8, 9, 2, 7] is fundamentally different from the problem considered here in which each agent can only travel in a certain subgraph: In


33rd International Symposium on Algorithms and Computation (ISAAC 2022).
Editors: Sang Won Bae and Heejin Park; Article No. 49; pp. 49:1–49:16
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

our problem, an agent can still travel an arbitrary distance by moving back and forth many times within its subgraph. Furthermore, the subgraph in which an agent can travel cannot necessarily be defined via a budget constraint. This means that neither hardness results nor algorithmic results translate directly between the two problems.

As motivation for considering agents with movement restrictions, we note that in a realistic setting, the usage of drones may be regulated by licenses that forbid some drones from flying in certain areas. The license of a drone operator may only allow that operator to cover a certain area. Furthermore, densely populated areas may have restrictions on which drones are allowed to operate there. Package delivery with multiple collaborating agents might also involve different types of agents (boats, cars, flying drones, etc.), where it is natural to consider the case that each agent can traverse only a certain part of the graph: For example, a boat can only traverse edges that represent waterways.

In our setting, we are given an undirected graph $G = (V, E)$ with a source node $s$ and a target node $y$ of the package as well as a set of $k$ agents. The subgraph in which an agent $a$ is allowed to operate is denoted by $G_a = (V_a, E_a)$. Each agent can pick up the package at its source location or from another drone, and it can deliver the package to the target location or hand it to another drone. We consider the objective of minimizing the delivery time, i.e., the time when the package reaches $y$, as well as the objective of minimizing the total energy consumption of the drones. We only consider the problem for a single package.

**Related work.** Collaborative delivery of a package from a source node $s$ to a target node $y$ with the goal of minimizing the delivery time was considered by Bärtschi et al. [4]. For $k$ agents in a graph with $n$ nodes and $m$ edges, they showed that the problem can be solved in $O(k^2m + kn^2 + \text{APSP})$ time for a single package, where APSP is the time for computing all-pairs shortest paths in a graph with $n$ nodes and $m$ edges. Carvalho et al. [6] improved the time complexity for the problem to $O(kn \log n + km)$ and showed that the problem is *NP*-hard if two packages must be delivered.

For minimizing the energy consumption for the delivery of a package, Bärtschi et al. [3] gave a polynomial algorithm for one package and showed that the problem is *NP*-hard for several packages.

The combined optimization of delivery time $\mathcal{T}$ and energy consumption $\mathcal{E}$ for the collaborative delivery of a single package has also been considered. Lexicographically minimizing $(\mathcal{E}, \mathcal{T})$ can be done in polynomial time [5], but lexicographically minimizing $(\mathcal{T}, \mathcal{E})$ or minimizing any linear combination of $\mathcal{T}$ and $\mathcal{E}$ is *NP*-hard [4].

Delivering a package using $k$ energy-constrained agents has been shown to be strongly *NP*-hard in general graphs [8] and weakly *NP*-hard on a path [9]. Bärtschi et al. [2] showed that the variant where each agent must return to its initial location can be solved in polynomial time for tree networks. The variant in which the package must travel via a fixed path in a general graph has been studied by Chalopin et al. [7].

**Our results.** In Section 2, we introduce definitions and give some auxiliary results. In Section 3, we show that movement restrictions make the drone delivery problem harder for both objectives: For minimizing the delivery time, we show that the problem is *NP*-hard to approximate within ratio $O(n^{1-\epsilon})$ or $O(k^{1-\epsilon})$. For minimizing the energy consumption, we show *NP*-hardness. These results hold even if all agents have the same speed and the same energy consumption rate.

In Section 4, we propose an $O(\min\{n, k\})$-approximation algorithm for the problem of minimizing the delivery time. The algorithm first computes a schedule with minimum delivery time for the problem variant where an arbitrary number of copies of each agent is available.

Then it transforms the schedule into one that is feasible for the problem with a single copy of each agent. The algorithm can also handle handovers on edges. In Section 5, we give a 2-approximation algorithm for the problem of minimizing the total energy consumption. We again first compute an optimal schedule that may use several copies of each agent and then transform the schedule into one with a single copy of each agent.

In Section 6, we first consider the special case where the graph is a path (line) and the subgraph of each agent is a subpath. For this case, we show that the problem of minimizing the delivery time is *NP*-hard if the agents have different speeds. If the agents have the same speed or the same energy consumption rate, we show that the problem of minimizing the delivery time and the problem of minimizing the total energy consumption are both polynomial-time solvable even for the more general case when the graph is a tree, or when the graph is arbitrary but the subgraph of every agent is isometric (defined in Section 6). Conclusions are presented in Section 7. Proofs omitted due to space restrictions can be found in the full version [11].

## 2 Preliminaries

We now define the *drone delivery* (DD) problem formally. We are given a connected graph $G = (V, E)$ with edge lengths $\ell : E \to \mathbb{R}_{\geq 0}$, where $\ell(u, v)$ represents the distance between $u$ and $v$ along edge $\{u, v\}$. (We sometimes write $G = (V, E, \ell)$ to include an explicit reference to $\ell$.) Let $n = |V|$ and $m = |E|$. We are also given a set $A$ containing $k \geq 1$ mobile agents (representing drones). Each agent $a \in A$ is specified by $a = (p_a, v_a, w_a, V_a, E_a)$, where $p_a \in V$ is the agent's *initial position*, and $v_a > 0$ and $w_a \geq 0$ are the agent's *velocity* (or speed) and *energy consumption rate*, respectively. To traverse a path of length $x$, agent $a$ takes time $x/v_a$ and consumes $x \cdot w_a$ units of energy. Furthermore, $V_a \subseteq V$ and $E_a \subseteq E$ are the *node-range* and *edge-range* of agent $a$, respectively. Agent $a$ can only travel to/via nodes in $V_a$ and edges in $E_a$. We require $p_a \in V_a$. To ensure meaningful solutions, we make the following two natural assumptions:

- For each agent $a$, the graph $G_a = (V_a, E_a)$ is a connected subgraph of $G$.
- The union of the subgraphs $G_a$ over all agents is the graph $G$, i.e., $\bigcup_{a\in A} V_a = V$ and $\bigcup_{a\in A} E_a = E$. This implies that there is a feasible schedule for any package $(s, y)$.

The package is specified by $(s, y)$, where $s, y \in V$ are the *start node* (start location) and *destination node* (target location), respectively. The task is to find a schedule for delivering the package from the start node to the destination node while achieving a specific objective. The problem of minimizing the delivery time is denoted by DDT, and the problem of minimizing the consumption by DDC.

To describe solutions of the problems, we need to define how a *schedule* can be represented. A schedule is given as a list of *trips* $\mathcal{S} = \{S_1, S_2, \ldots, S_h\}$:

$$\{(a_1, t_1, \langle o_1, \ldots, u_0\rangle, \langle u_0, \ldots, u_1\rangle), \ldots, (a_h, t_h, \langle o_h, \ldots, u_{h-1}\rangle, \langle u_{h-1}, \ldots, u_h\rangle)\}$$

The $i$-th trip $S_i = (a_i, t_i, \langle o_i, \ldots, u_{i-1}\rangle, \langle u_{i-1}, \ldots, u_i\rangle)$ represents two consecutive trips taken by agent $a_i$ starting at time $t_i \geq 0$: an *empty movement* trip traversing nodes $O_i = \langle o_i, \ldots, u_{i-1}\rangle$, and a *delivery trip* (during which $a_i$ carries the package) of traversing nodes $U_i = \langle u_{i-1}, \ldots, u_i\rangle$. The agent $a_i$ picks up the package at node $u_{i-1}$ and drops it off at node $u_i$. With slight abuse of notation, we also use $O_i$ and $U_i$ to denote the set of edges in each of these two trips. If $S_i$ is the first trip of agent $a_i$, then $o_i$ is the agent's initial location. Otherwise, $o_i$ is the location where the agent dropped off the package at the end of its previous trip. Initially, each agent $a \in A$ is at node $p_a$ at time 0. In the definition of schedules, when we allow two agents to meet on an edge to hand over the package, we allow the nodes also to be points on edges: For example, a node $u_i$ could be the point on an edge $\{u, v\}$ with length 5 that is at distance 2 from $u$ (and hence at distance 3 from $v$).

Let $T(u_i)$ (resp. $C(u_i)$) denote the time passed (resp. the energy consumed) until the package is delivered to node $u_i$ in schedule $\mathcal{S}$, i.e.,

$$T(u_i) = \max \left\{ T(u_{i-1}), t_i + \sum_{e \in O_i} \frac{\ell(e)}{v_{a_i}} \right\} + \sum_{e \in U_i} \frac{\ell(e)}{v_{a_i}},$$

$$C(u_i) = C(u_{i-1}) + w_{a_i} \cdot \sum_{e \in O_i} \ell(e) + w_{a_i} \cdot \sum_{e \in U_i} \ell(e).$$

The pick-up location of the first agent must be the start node, i.e., $u_0 = s$, and the drop-off location of the last agent must be the destination node, $u_h = y$. We let $T(u_0) = 0$ and $C(u_0) = 0$. The goal of the DDT problem is to find a feasible schedule $\mathcal{S}$ that minimizes the delivery time $T(y)$, and the goal of the DDC problem is to find a feasible schedule $\mathcal{S}$ that minimizes the energy consumption $C(y)$.

So far, we have defined the DDT and DDC problem. As in previous studies [6], we further distinguish variants of these problems based on the *handover manner*.

**Handover manner.** The handover of the package between two agents may occur at a node or at some interior point of an edge. When the handovers are restricted to be on nodes, we say the drone delivery problem is handled with *node handovers*, and when the handover can be done on a node or at an interior point of an edge, we say that the drone delivery problem is handled with *edge handovers*. We use the subscripts N and E to represent node handovers and edge handovers, respectively. Thus, we get four variants of the drone delivery problem: $\mathsf{DDT_N}$, $\mathsf{DDT_E}$, $\mathsf{DDC_N}$ and $\mathsf{DDC_E}$.

**With or without initial positions.** We additionally consider problem variants in which the initial positions of the agents are not fixed (given), which means that the initial positions $p_a$ for $a \in A$ can be chosen by the algorithm. When the initial positions are fixed, we say that the problem is *with initial positions*, and when the initial positions are not fixed, we say that the problem is *without initial positions*. When we do not specify that a problem is without initial positions, we always refer to the problem with initial positions by default.

In the rest of the paper, in order to simplify notation, the $i$-th trip in a schedule $\mathcal{S}$ is usually written in simplified form as $(a_i, u_{i-1}, u_i)$, where $a_i$ is the agent, $u_{i-1}$ is the pick-up location of the package, and $u_i$ is the drop-off location of the package. We can omit the agent's empty movement trip (including its previous position) and its start time because the agent $a_i$ always takes the path in $G_{a_i} = (V_{a_i}, E_{a_i})$ with minimum cost (travel time or energy consumption) from its previous position to the current pick-up location $u_{i-1}$ and then from $u_{i-1}$ to $u_i$.

For each node $v \in V$, we use $A(v)$ to denote the set of agents that can visit the node $v$; and for every edge $\{u, v\}$ in $E$, we use $A(u, v)$ to denote the set of agents that can traverse the edge $\{u, v\}$. For any pair of nodes $u, v \in V_a$ for some $a \in A$, we denote by $\mathsf{dist}_a(u, v)$ the length of the shortest path between node $u$ and $v$ in the graph $G_a = (V_a, E_a)$.

**Useful properties.** We show some basic properties for the drone delivery problem. The following lemma can be shown by an exchange argument: If an agent was involved in the package delivery at least twice, letting that agent keep the package from its first involvement to its last involvement does not increase the delivery time nor the energy consumption.

▶ **Lemma 1.** *For every instance of* $\mathsf{DDT_N}$*,* $\mathsf{DDT_E}$*,* $\mathsf{DDC_N}$ *and* $\mathsf{DDC_E}$ *(with or without initial positions), there is an optimal solution in which each of the involved agents picks up and drops off the package exactly once.*

For $\mathsf{DDC}$, we can show that handovers at interior points of edges cannot reduce the energy consumption. If two agents carry the package consecutively over parts of the same edge, letting one of the two agents carry the package over those parts can be shown not to increase the energy consumption.

▶ **Lemma 2.** *For any instance of* $\mathsf{DDC_N}$ *and* $\mathsf{DDC_E}$ *(with or without initial positions), there is a solution that is simultaneously optimal for both problems. In other words, there is an optimal solution for* $\mathsf{DDC_E}$ *in which all handovers of the package take place at nodes of the graph.*

For the case that $G_a = G$ for all $a \in A$, it has been observed in previous work [4, 6] that there is an optimal schedule for $\mathsf{DDT_N}$ and $\mathsf{DDT_E}$ in which the velocities of the involved agents are strictly increasing, and an optimal schedule for $\mathsf{DDC_N}$ and $\mathsf{DDC_E}$ in which the consumption rates of the involved agents are strictly decreasing. We remark that this very useful property does not necessarily hold in our setting with agent movement restrictions. This is the main reason why the problem becomes harder, as shown by our hardness results in Section 3 and, even for path networks, in Section 6.

## 3 Hardness results

In this section, we prove several hardness results for $\mathsf{DDT}$ and $\mathsf{DDC}$ via reductions from the *NP*-complete 3-*dimensional matching problem* (3DM) [12]. All of them apply to the case where all agents have the same speed and the same energy consumption rate. We first present the construction of a *base instance* of $\mathsf{DDT}$, which we then adapt to obtain the hardness results for $\mathsf{DDT}$ and $\mathsf{DDC}$. The instances that we create have the property that every edge $e$ of the graph is only in the set $E_a$ of a single agent $a$. Therefore, handovers on edges are not possible for these instances, and so the variants of the problems with node handovers and with edge handovers are equivalent for these instances. Hence, we only need to consider the problem variant with node handovers in the proofs.

The problem 3DM is defined as follows: Given are three sets $X, Y, Z$, each of size $n$, and a set $\mathcal{F} \subseteq X \times Y \times Z$ consisting of $m$ triples. Each triple in $\mathcal{F}$ is of the form $(x, y, z)$ with $x \in X$, $y \in Y$, and $z \in Z$. The question is: Is there a set of $n$ triples in $\mathcal{F}$ such that each element of $X \cup Y \cup Z$ is contained in exactly one of these $n$ triples?

Let an instance of 3DM be given by $X, Y, Z$ and $\mathcal{F}$. Assume $X = \{x_1, x_2, \dots, x_n\}$, $Y = \{y_1, y_2, \dots, y_n\}$ and $Z = \{z_1, z_2, \dots, z_n\}$. Number the triples in $\mathcal{F}$ from 1 to $m = |\mathcal{F}|$ arbitrarily, and let the $j$-th triple be $t_j = (x_{f(j)}, y_{g(j)}, z_{h(j)})$ for suitable functions $f, g, h : [m] \to [n]$. We create from this a base instance $I$ of $\mathsf{DDT_N}$ that consists of $n$ selection gadgets and $3n$ agent gadgets. Carrying the package through a selection gadget corresponds to selecting one of the $m$ triples of $\mathcal{F}$. The $n$ selection gadgets are placed consecutively, so that the package travels through all of them (unless it makes a detour that increases the delivery time). For each element $q$ of $X \cup Y \cup Z$, there is an agent gadget containing the start position of a unique agent. The agent gadget for element $q$ ensures that the agent can carry the package on the edge corresponding to element $q$ on a path through a selection gadget if and only if that path corresponds to a triple that contains $q$. If the instance of 3DM is a yes instance, then each of the agents from the agent gadgets only needs to transport

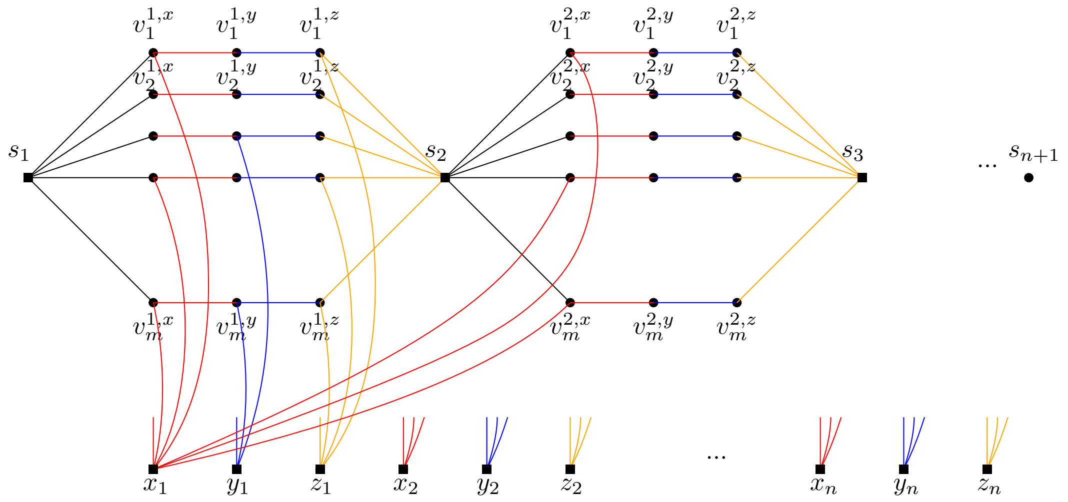


**Figure 1** The graph $\bar{G} = (\bar{V}, \bar{E})$ used for the inapproximability and hardness proofs for DDT and DDC. The inner edges have length 0, and the outer edges have length $M$. Each square node represents a drone's initial location. Each colored (red, blue, and orange) path represents a set of edges that can be visited by the same agent which is initially located at a location in $\{x_i, y_i, z_i \mid i \in [n]\}$. For each $i \in [n]$, there is an additional agent that is initially located at $s_i$ and which can visit edges $\{\{s_i, v_x^{i,x}\} \mid j \in [m]\}$.

the package on one edge. Otherwise, an agent must travel (with or without the package) from one selection gadget to another one, and the extra time consumed by this movement increases the delivery time.

Formally, the instance of $\mathsf{DDT_N}$ with graph $\bar{G} = (\bar{V}, \bar{E})$ is created as follows. See Figure 1 for an illustration. The vertex set $\bar{V}$ contains $3n$ *agent nodes* and $n(3m+1)+1$ nodes in selection gadgets, a total of $4n + 3mn + 1$ nodes. The agent nodes comprise a node $x_i$ for each $x_i \in X$, a node $y_i$ for each $y_i \in Y$, and a node $z_i$ for each $z_i \in Z$. The node set of selection gadget $i$, for $1 \le i \le n$, is $\{s_i, s_{i+1}\} \cup \{v_j^{i,x}, v_j^{i,y}, v_j^{i,z} \mid 1 \le j \le m\}$, a set containing $3m+2$ nodes. For $1 \le i < n$, the node $s_{i+1}$ is contained in selection gadget $i$ and in selection gadget $i+1$. The path through the three nodes $v_j^{i,x}, v_j^{i,y}, v_j^{i,z}$ (in selection gadget $i$ for any $i$) represents the triple $t_j$ in $\mathcal{F}$. The package is initially located at node $s_1$ and must be delivered to node $s_{n+1}$.

The edge set $\bar{E}$ contains two types of edges: *inner* edges, each of which connects two nodes in a selection gadget, and *outer* edges, each of which connects an agent node and a node in a selection gadget. For every $i \in [n]$ and $j \in [m]$, there are four inner edges $\{s_i, v_j^{i,x}\}$, $\{v_j^{i,x}, v_j^{i,y}\}$, $\{v_j^{i,y}, v_j^{i,z}\}$ and $\{v_j^{i,z}, s_{i+1}\}$ of length 0.[1] So there are $4mn$ inner edges. The outer edges are as follows: For every $i \in [n]$ and $j \in [m]$, there are the three outer edges $\{x_{f(j)}, v_j^{i,x}\}$, $\{y_{g(j)}, v_j^{i,y}\}$, $\{z_{h(j)}, v_j^{i,z}\}$ of length $M$ for some $M > 0$. (We can set $M = 1$.)

This completes the description of the graph $\bar{G} = (\bar{V}, \bar{E})$ with edge lengths.

Now we define the set of agents. We have $4n$ agents with unit speed and unit energy consumption rate. Their initial locations are $\{x_i, y_i, z_i, s_i \mid i \in [n]\}$. The agents with initial location in $\{x_i, y_i, z_i \mid i \in [n]\}$ are called *element agents*. For each $i \in [n]$, the node ranges and edge ranges of the agents with initial location with subscript $i$ are defined as follows:

[1] We could set the lengths of these edges to a small $\epsilon > 0$ if we wanted to avoid edges of length 0.

- The agent located at $x_i$ has node range $\{x_i, v_j^{i',x}, v_j^{i',y} \mid i' \in [n], j \in [m], f(j) = i\}$ and edge range $\{\{x_i, v_j^{i',x}\}, \{v_j^{i',x}, v_j^{i',y}\} \mid i' \in [n], j \in [m], f(j) = i\}$.
- The agent located at $y_i$ has node range $\{y_i, v_j^{i',y}, v_j^{i',z} \mid i' \in [n], j \in [m], g(j) = i\}$ and edge range $\{\{y_i, v_j^{i',y}\}, \{v_j^{i',y}, v_j^{i',z}\} \mid i' \in [n], j \in [m], g(j) = i\}$.
- The agent located at $z_i$ has node range $\{z_i, v_j^{i',z}, s_{i'+1} \mid i' \in [n], j \in [m], h(j) = i\}$ and edge range $\{\{z_i, v_j^{i',z}\}, \{v_j^{i',z}, s_{i+1}\} \mid i' \in [n], j \in [m], h(j) = i\}$.
- The agent located at $s_i$ has node range $\{s_i, v_j^{i,x} \mid j \in [m]\}$ and edge range $\{\{s_i, v_j^{i,x}\} \mid j \in [m]\}$.

We now show that the given instance of 3DM is a yes-instance if and only if the constructed base instance of $\mathsf{DDT_N}$ has an optimal schedule with delivery time $M$. First, assume that the given instance of 3DM is a yes-instance, and let $\{t_{k_1}, t_{k_2}, \ldots, t_{k_n}\} \subseteq \mathcal{F}$ be a perfect matching. We let the package travel from $s_1$ to $s_{n+1}$ via the $n$ selection gadgets, using the path via nodes $v_{k_i}^{i,x}, v_{k_i}^{i,y}, v_{k_i}^{i,z}$ in selection gadget $i$, for $i \in [n]$. This path consists of $4n$ edges. Each of the $4n$ agents needs to carry the package on exactly one edge of this path. All the element agents reach the node where they pick up the package at time $M$. As all edges in the selection gadgets have length 0, this shows that the package reaches $s_{n+1}$ at time $M$. It is clear that this solution is optimal because at least one element agent must take part in the delivery and cannot pick up the package before time $M$.

Now, assume that the base instance of $\mathsf{DDT_N}$ admits a schedule with delivery time $M$. It is not hard to see that the schedule must be of the above format, using each agent on exactly one edge in one selection gadget. This is because it takes time $2M$ for an element agent to move from one selection gadget to another one via its initial location.

This shows that there is a schedule with delivery time $M$ if and only if the given instance of 3DM is a yes-instance. Furthermore, as any element agent needs time $M$ to reach a pick-up point in a selection gadget and time $2M$ to move to a different selection gadget (with or without the package), it is clear that the optimal schedule will have length at least $3M$ if the given instance of 3DM is a no-instance. This already shows that $\mathsf{DDT_N}$ is *NP*-hard and does not admit a polynomial-time approximation algorithm with approximation ratio smaller than 3 unless $P = NP$, but we can strengthen the inapproximability result by concatenating $q = (4n+3mn)^c$ copies of the base instance (identifying node $s_{n+1}$ of one copy with node $s_1$ of the next copy) for an arbitrarily large constant $c$ and letting the package be transported from $s_1$ in the first copy to $s_{n+1}$ in the last copy. If the given instance of 3DM is a yes-instance, the delivery time is still $M$, but if it is a no-instance, it is $M + 2Mq = M(2q + 1)$.

▶ **Theorem 3.** *For any constant $\epsilon > 0$, it is NP-hard to approximate $\mathsf{DDT_N}$ (or $\mathsf{DDT_E}$) within a factor of $O(\min\{n^{1-\epsilon}, k^{1-\epsilon}\})$ even if all agents have the same speed.*

To show *NP*-hardness for $\mathsf{DDC}$, we adapt the base instance by numbering the columns of the selection gadgets to which outer edges are attached from 1 to $3n$ (from left to right) and letting the outer edges attached to column $i$ have length $2^{3n+1-i}$. If the given instance of 3DM is a yes-instance, exactly one outer edge attached to each column will be used, for a total energy consumption of $2^{3n+1} - 2$. Otherwise, some column will be the first column in which an outer edge is used twice, and the total energy consumption will be at least $2^{3n+1}$.

▶ **Theorem 4.** *The problems $\mathsf{DDC_N}$ (and $\mathsf{DDC_E}$) are NP-hard even if all agents have the same energy consumption.*

Finally, for the problem variants without initial positions, we observe that the base instance admits a delivery schedule with delivery time 0 and energy consumption 0 if and only if the given instance of 3DM is a yes-instance. This shows that there cannot be any polynomial-time approximation algorithm for any of the DDT and DDC problem variants without initial positions. This holds since we allow zero-length edges. If we were to require strictly positive edge lengths, it would be possible to obtain approximation algorithms with ratios that depend on the ratio of maximum to minimum edge length.

## 4 Approximation algorithm for the DDT problem

We first present an optimal algorithm for $\mathsf{DDT_N}$ under the assumption that there are as many copies of each agent as we need. We start by introducing some notation used in the algorithm. For any edge $\{u, v\} \in E_a$ for some $a \in A$, we denote by $\mathsf{eT}_a(v, u \prec v)$ the earliest time for the package to arrive at $v$ if the package is carried over the edge $\{u, v\}$ by a copy of agent $a$. In addition, we use $\mathsf{eT}(v, u \prec v)$ to denote the earliest time for the package to arrive at node $v$ if the package is carried over the edge $\{u, v\}$ by some agent, i.e., $\mathsf{eT}(v, u \prec v) = \min\{\mathsf{eT}_a(v, u \prec v) \mid \{u, v\} \in E_a, a \in A\}$. For every $v \in V$, we use $\mathsf{eT}(v)$ to denote the earliest arrival time for the package at node $v$, i.e., $\mathsf{eT}(v) = \min\{\mathsf{eT}(v, u \prec v) \mid \{u, v\} \in E\}$. Note that the package is initially at location $s$ at time 0, i.e., $\mathsf{eT}(s) = 0$. Given a node $v$, we denote by $\mathcal{S}(v)$ the schedule for carrying the package from $s$ to $v$, i.e., $\mathcal{S}(v) = \{(a_1, s, u_1), (a_2, u_1, u_2), \ldots, (a_h, u_{h-1}, v)\}$.

Our algorithm adapts the approach of a time-dependent Dijkstra's algorithm [4, 6]. Algorithm 1 shows the pseudo-code. For each node $v \in V$, we maintain a value $\mathsf{eT}(v)$ that represents the current upper bound on the earliest time when the package can reach $v$ (line 3–4). Initially, we set the earliest arrival time for $s$ to 0 and for all other nodes to $\infty$. We maintain a priority queue $Q$ of nodes $v$ with finite $\mathsf{eT}(v)$ value that have not been processed yet (line 8). In each iteration of the while-loop, we process a node $u$ in $Q$ having the earliest arrival time (line 10), where $u = s$ in the first iteration. For each unprocessed neighbor node $v$ of $u$, we calculate the earliest arrival time $\mathsf{eT}(v, u \prec v)$ at node $v$ if the package is carried over the edge between $u$ and $v$ by some agent (and we store the identity of that agent in $L(v, u \prec v)$), by calling the subroutine NeiDelivery$(u, v, t)$ (Algorithm 2). If this earliest arrival time is smaller than $\mathsf{eT}(v)$, then $\mathsf{eT}(v)$ is updated and $v$ is inserted into $Q$ (if it is not yet in $Q$) or its priority reduced to the new value of $\mathsf{eT}(v)$. The algorithm terminates and returns the value $\mathsf{eT}(y)$ when the node being processed is $y$ (line 12). The schedule $\mathcal{S} = \mathcal{S}(y)$ can be constructed in a backward manner because we store for each node $v$ the involved agent $L(v)$ and its predecessor node $\mathsf{prev}(v)$ (line 20 and 21).

To compute $\mathsf{eT}(v, u \prec v)$ in line 17 of Algorithm 1, we call NeiDelivery$(u, v, t)$ (Algorithm 2): That subroutine calculates for each agent $a \in A$ with $\{u, v\} \in E_a$, i.e., for all $a \in A(u, v)$, the time when the package reaches $v$ if that agent picks it up at $u$ and carries it over $\{u, v\}$ to $v$. The earliest arrival time at $v$ via the edge $\{u, v\}$ is returned as $\mathsf{eT}(v, u \prec v)$, and the agent $a^*$ achieving that arrival time is returned as $L(v, u \prec v)$.

▶ **Lemma 5.** *The following statements hold for the schedule found by Algorithm 1.*

**(i)** *It may happen that an agent picks up and drops off the package more than once. Each time an agent $a$ carries the package over a path of consecutive edges, a copy of the agent starts at $p_a$, travels to the start node $u$ of the path, picks up the package at time $\max\{\mathsf{eT}(u), \frac{\mathsf{dist}_a(p_a,u)}{v_a}\}$, and then carries the package over the edges of the path.*

**(ii)** *The package is carried to each node $v \in V$ at most once, and thus the schedule carries the package over at most $|V| - 1$ edges.*

▶ **Lemma 6.** *There is an algorithm that computes an optimal schedule in time $O(k(m + n \log n))$ for $\mathsf{DDT_N}$ under the assumption that an arbitrary number of copies of each agent can be used. The package gets delivered from $s$ to $y$ along a simple path with at most $|V| - 1$ edges.*

**Algorithm 1** Algorithm for DDT.

**Data:** Graph $G = (V, E, \ell)$; package source node $s$ and target node $y$; agent $a$ with velocity $v_a$ and initial location $p_a$ for $a \in A$
**Result:** earliest arrival time for package at target location $y$, i.e., $\mathsf{eT}(y)$

1 **begin**
2   compute $\mathsf{dist}_a(p_a, v)$ for $a \in A$ and $v \in V_a$
3   $\mathsf{eT}(s) \leftarrow 0$
4   $\mathsf{eT}(v) \leftarrow \infty$ for all $v \in V \setminus \{s\}$
5   $L(v) \leftarrow \emptyset$ for all $v \in V$ `// agent bringing the package to` $v$
6   $\mathsf{proc}(v) \leftarrow 0$ for all $v \in V$ `// all nodes` $v$ `are unprocessed`
7   $\mathsf{prev}(v) \leftarrow \emptyset$ for all $v \in V$ `// previous node on optimal package path to` $v$
8   $Q \leftarrow \{s\}$ `// priority queue of pending nodes`
9   **while** $Q \neq \emptyset$ **do**
10     $u \leftarrow \arg\min\{\mathsf{eT}(v) \mid v \in Q\}$ `// node with minimum arrival time in` $Q$
11     $Q \leftarrow Q \setminus \{u\}$ and $\mathsf{proc}(u) \leftarrow 1$
12     **if** $u = y$ **then**
13       break
14     **end**
15     $t \leftarrow \mathsf{eT}(u)$ `// arrival time when package reaches` $u$
16     **for** *neighbors* $v$ *of* $u$ *with* $\mathsf{proc}(v) = 0$ *and* $A(u, v) \neq \emptyset$ **do**
17       $\{\mathsf{eT}(v, u \prec v), L(v, u \prec v)\} \leftarrow \textsc{NeiDelivery}(u, v, t)$
18       **if** $\mathsf{eT}(v, u \prec v) < \mathsf{eT}(v)$ **then**
19         $\mathsf{eT}(v) \leftarrow \mathsf{eT}(v, u \prec v)$
20         $L(v) \leftarrow L(v, u \prec v)$
21         $\mathsf{prev}(v) \leftarrow \{u\}$
22         **if** $v \notin Q$ **then**
23           $Q \leftarrow Q \cup \{v\}$ with earliest arrival time $\mathsf{eT}(v)$
24         **end**
25       **end**
26     **end**
27   **end**
28   **return** $\mathsf{eT}(y)$
29 **end**

**Proof.** We claim that the arrival time $\mathsf{eT}(u)$ for each node $u$ is minimum by the time $u$ gets processed. Obviously, the first processed node $s$ has arrival time $\mathsf{eT}(s) = 0$. Whenever a node $u$ is processed, the earliest arrival time for each unprocessed neighbor is updated (line 19 of Algorithm 1) if the package can reach that neighbor earlier via node $u$ (line 4 in Algorithm 2 identifies the agent $a^*$ that can bring the package from $u$ to the neighbor the fastest). At the time when a node $u$ is removed from the priority queue $Q$ and starts to be processed, all nodes $v$ with $\mathsf{eT}(v) < \mathsf{eT}(u)$ have already been processed, and so its value $\mathsf{eT}(u)$ must be equal to the earliest time when the package can reach $u$. The algorithm terminates when $y$ is removed from $Q$.

In lines 19–21 of Algorithm 1, we update the arrival time $\mathsf{eT}(v)$ and the agent $L(v)$ as well as the predecessor node $\mathsf{prev}(v)$ without explicitly maintaining the schedules $\mathcal{S}(v)$. The schedule $\mathcal{S}(v)$ can be retraced from $L(\cdot)$ and $\mathsf{prev}(\cdot)$ since the schedule found for $\mathsf{eT}(v)$ visits each node in $V$ at most once, cf. Lemma 5. This also shows that the package gets delivered from $s$ to $y$ along a simple path (with at most $|V| - 1$ edges) in $G$.

**Algorithm 2** Algorithm NeiDelivery$(u, v, t)$ for $\mathsf{DDT_N}$.

---

**Data:** Edge $\{u, v\}$, arrival time for node $u$ $\mathsf{eT}(u) = t$, agents $A(u, v)$
**Result:** $\mathsf{eT}(v, u \prec v)$
**for** $a \in A(u, v)$ **do**
  $\mathsf{eT}_a(v, u \prec v) = \max\{t, \frac{\mathsf{dist}_a(p_a,u)}{v_a}\} + \frac{\ell(u,v)}{v_a}$
**end**
$a^* = \arg\min\{\mathsf{eT}_a(v, u \prec v) \mid a \in A(u, v)\}$;
$\mathsf{eT}(v, u \prec v) \leftarrow \mathsf{eT}_{a^*}(v, u \prec v)$
$L(v, u \prec v) \leftarrow a^*$
**return** $\mathsf{eT}(v, u \prec v)$ and $L(v, u \prec v)$

---

We can analyze the running-time of Algorithm 1 as follows. The distances $\mathsf{dist}_a(p_a, v)$ can be pre-computed by running Dijkstra's algorithm with Fibonacci heaps in time $O(m + n \log n)$ [10] for each agent $a$ with source node $p_a$ in the graph $G_a$, taking total time $O(k(m + n \log n))$ (line 2). Selecting and removing a minimum element from the priority queue $Q$ in lines 10–11 takes time $O(\log n)$. At most $n$ nodes will be added into $Q$ (and later removed from $Q$), so the running time for inserting and removing elements from $Q$ is $O(n \log n)$. For each processed node $u$, we compute the value $\mathsf{eT}(v, u \prec v)$ for each unprocessed adjacent node $v$ with $A(u, v) \neq \emptyset$ in time $O(|A(u, v)|)$. Overall we get a running time of $O(k(m + n \log n) + n \log n + km) = O(k(m + n \log n))$. ◀

Next, we can show how to convert the delivery schedule $\mathcal{S}$ with delivery time $\mathcal{T}$ produced by the algorithm of Lemma 6 into a schedule that is feasible with a single copy per agent, and we bound the resulting increase in the delivery time $\mathcal{T}$ to obtain an approximation algorithm for $\mathsf{DDT_N}$. The conversion consists of the repeated application of modification steps. Each step considers the first agent $a$ that is used at least twice. Let $a$ pick up the package at $u_{i-1}$ in its first trip and carry it from $u_{j-1}$ to $u_j$ in its last trip. We then modify the schedule so that $a$ picks up the package at $u_{i-1}$ and carries it all the way to $u_j$ along a shortest path in $G_a$. We have $\frac{1}{v_a}\mathsf{dist}_a(p_a, u_{i-1}) \leq \mathcal{T}$ and $\frac{1}{v_a}(\mathsf{dist}_a(p_a, u_{j-1}) + \mathsf{dist}_a(u_{j-1}, u_j)) \leq \mathcal{T}$. By the triangle inequality, $\mathsf{dist}_a(u_{i-1}, u_j) \leq \mathsf{dist}_a(u_{i-1}, p_a) + \mathsf{dist}_a(p_a, u_{j-1}) + \mathsf{dist}_a(u_{j-1}, u_j)$. Hence, agent $a$ needs time at most $2\mathcal{T}$ to carry the package from $u_{i-1}$ to $u_j$, and so the delivery time increases by at most $2\mathcal{T}$. Furthermore, we can bound the number of modification steps by $\min\{\frac{n-1}{3}, k - 1\}$.

▶ **Theorem 7.** *There is a* $\min\{2n/3 + 1/3, 2k - 1\}$*-approximation algorithm for* $\mathsf{DDT_N}$.

To adapt the approach from $\mathsf{DDT_N}$ to $\mathsf{DDT_E}$, we can adapt the algorithm of Lemma 6 to edge handovers by using as a subroutine the FastLineDelivery$(u, v, t)$ method from [6], which calculates in $O(k \log k)$ time an optimal delivery schedule using the agents in $A(u, v)$ to transport the package that arrives at $u$ at time $t$ from $u$ to $v$ over the edge $\{u, v\}$. When transforming the resulting package delivery schedule into one that uses each agent at most once, the number of modification steps can be bounded by $\min\{n - 1, k - 1\}$.

▶ **Theorem 8.** *There is a* $\min\{2n - 1, 2k - 1\}$*-approximation algorithm for* $\mathsf{DDT_E}$.

## 5 Approximation algorithm for the DDC problem

By Lemma 2, handovers on interior points of edges are not needed for DDC, so the results for $\mathsf{DDC_N}$ that we present in this section automatically apply to $\mathsf{DDC_E}$ as well. Therefore, we only consider $\mathsf{DDC_N}$ in the proofs. We first give an algorithm that solves $\mathsf{DDC_N}$ optimally if there is a sufficient number of copies of every agent.

Let an instance of $\mathsf{DDC_N}$ be given by a graph $G = (V, E, \ell)$, package start node $s$ and destination node $y$, and a set $A$ of $k$ agents where $a = (p_a, v_a, w_a, V_a, E_a)$ for $a \in A$. We create a directed graph $G'$ in which a shortest path from $s'$ to $y'$ corresponds to an optimal delivery schedule. This approach is motivated by a method used by Bärtschi et al. [3]. We construct the directed edge-weighted graph $G' = (V', E', \ell')$ as follows:

- For each node $u \in V$ and each agent $a \in A$ with $u \in V_a$, create a node $u_a$ in $V'$. In addition, add a node $s'$ and a node $y'$.
- For all $a \in A$ with $s \in V_a$, add an arc $(s', s_a)$ with $\ell'(s', s_a) = w_a \cdot \mathsf{dist}_a(p_a, s)$. For all $a \in A$ with $y \in V_a$, add an arc $(y_a, y')$ with $\ell'(y_a, y') = 0$.
- For $\{u, x\} \in E$, for each $a$ with $\{u, x\} \in E_a$, create two arcs $(u_a, x_a)$ and $(x_a, u_a)$ with $\ell'(x_a, u_a) = \ell'(u_a, x_a) = w_a \cdot \ell(u, x)$.
- For $u \in V$ and agents $a, \bar{a} \in A(u)$, create the following two arcs: $(u_a, u_{\bar{a}})$ with $\ell'(u_a, u_{\bar{a}}) = w_{\bar{a}} \cdot \mathsf{dist}_{\bar{a}}(p_{\bar{a}}, u)$, and $(u_{\bar{a}}, u_a)$ with $\ell'(u_{\bar{a}}, u_a) = w_a \cdot \mathsf{dist}_a(p_a, u)$.

Intuitively, a node $u_a$ in $G'$ represents the agent $a$ carrying the package at node $u$ in $G$. An arc $(u_a, x_a)$ represent the agent $a$ carrying the package over edge $\{u, x\}$ from $u$ to $x$. An arc $(u_a, u_{\bar{a}})$ represents a copy of agent $\bar{a}$ traveling from $p_{\bar{a}}$ to $u$ and taking over the package from agent $a$ there. We can show that a shortest $s'$-$y'$ path in $G'$ corresponds to an optimal schedule with multiple copies per agent.

▶ **Lemma 9.** *An optimal schedule for* $\mathsf{DDC_N}$ *(and* $\mathsf{DDC_E}$*) can be computed in time* $O(nk^2 + n^2k)$ *under the assumption that an arbitrary number of copies of each agent can be used.*

**Proof.** We claim that a shortest $s'$-$y'$ path in $G'$ corresponds to an optimal delivery schedule. First, assume that an optimal delivery schedule $\mathcal{S}$ is

$$\{(a_1, s, u_1), (a_2, u_1, u_2), \ldots, (a_h, u_{h-1}, y)\},$$

where it is possible that $a_i = a_j$ for $j > i + 1$ because we allow copies of agents to be used. Then we can construct an $s'$-$y'$ path in $G'$ whose length equals the total energy consumption of $\mathcal{S}$ as follows: Start with the arc $(s', s_{a_1})$. Then use the arcs $(s_{a_1}, z^{(1)}_{a_1}), \ldots, (z^{(g)}_{a_1}, u_{1_{a_1}})$ corresponding to the path $s, z^{(1)}, \ldots, z^{(g)}, u_1$ (for some $g \geq 0$) along which agent $a_1$ carries the package from $s$ to $u_1$ in $\mathcal{S}$. Next, use the arc $(u_{1_{a_1}}, u_{1_{a_2}})$ representing the handover from $a_1$ to $a_2$ at $u_1$. Continue in this way until node $y_{a_h}$ is reached, and then follow the arc from there to $y'$. Similarly, any $s'$-$y'$ path $P$ in $G'$ can be translated into a delivery schedule in $G$ whose total energy consumption is equal to the length of $P$ in $G'$.

Finally, let us consider the running-time of the algorithm. First, we compute $\mathsf{dist}_a(p_a, v)$ for each agent $a$ and each node $v \in V$ by running Dijkstra's algorithm with Fibonacci heaps [10] once in $G_a$ with source node $p_a$ for each $a \in A$, taking time $O(k(n \log n + m)) = O(kn^2)$. The graph $G'$ has at most $n \cdot k + 2 \in O(nk)$ vertices and at most $2k + (n^2 \cdot k + n \cdot k^2) \in O(nk^2 + n^2k)$ arcs. It can be constructed in $O(nk^2 + n^2k)$ time as we have pre-computed the values $\mathsf{dist}_a(p_a, v)$. We can compute the shortest $s'$-$y'$ path in $G'$ in time $O(nk^2 + n^2k + nk\log(nk)) = O(nk^2 + n^2k)$ time using Dijkstra's algorithm. ◀

▶ **Theorem 10.** *There is a 2-approximation algorithm for* $\mathsf{DDC_N}$ *(and* $\mathsf{DDC_E}$*).*

**Proof.** Let an instance of $\mathsf{DDC_N}$ be given by a graph $G = (V, E, \ell)$, package start node $s$ and destination node $y$, and a set $A$ of $k$ agents where $a = (p_a, v_a, w_a, V_a, E_a)$ for $a \in A$. Compute an optimal delivery schedule that may use multiple copies of agents using the algorithm of Lemma 9. Then we transform the schedule into one that uses each agent at most once as follows: Let $a$ be the first agent that is used more than once in the delivery schedule.

Assume that $a$ carries the package from node $u$ to node $v$ during its first involvement in the delivery and from node $u'$ to node $v'$ during its last involvement in the delivery. The energy consumed by these two copies of agent $a$ is:

$$W_a = w_a(\mathsf{dist}_a(p_a, u) + \mathsf{dist}_a(u, v) + \mathsf{dist}_a(p_a, u') + \mathsf{dist}_a(u', v'))$$

We modify the schedule and let agent $a$ pick up the package at $u$ and carry it along a shortest path in $G_a$ from $u$ to $v'$. The trips in the original schedule that bring the package from $v$ to $u'$ are removed. The new energy consumption by agent $a$ is $w_a(\mathsf{dist}_a(p_a, u) + \mathsf{dist}_a(u, v'))$. By the triangle inequality, $\mathsf{dist}_a(u, v') \leq \mathsf{dist}_a(u, p_a) + \mathsf{dist}_a(p_a, u') + \mathsf{dist}_a(u', v')$. Hence, the new energy consumption is bounded by $2W_a$. As long as there is an agent that is used more than once, we apply the same modification step to the first such agent. The procedure terminates after at most $k-1$ modification steps. During the execution of the procedure, the energy consumption of every agent at most doubles. Therefore, the total energy consumption of the resulting schedule for $\mathsf{DDC_N}$ with a single copy of each agent is at most twice the total energy consumption of $\mathcal{S}$, which is a lower bound on the optimal energy consumption.

The algorithm of Lemma 9 takes $O(nk^2 + n^2k)$ time, which dominates the time needed to carry out the modification steps. Thus, the overall running time is $O(nk^2 + n^2k)$. ◀

## 6 Drone delivery on path and tree networks

In Section 6.1, we present hardness results for the drone delivery problems if the graph is a path. In Section 6.2, we show that the problems can be solved optimally in polynomial time for trees (and even for arbitrary graphs if the subgraphs of the agents satisfy a certain condition) provided that all agents have the same speed or the same energy consumption rate.

### 6.1 Hardness of DDT on the path

▶ **Theorem 11.** $\mathsf{DDT_N}$ *and* $\mathsf{DDT_E}$ *with initial positions are NP-complete if the given graph is a path.*

**Proof.** We give a reduction from the *NP*-complete Even-Odd Partition (EOP) problem [12] that is defined as follows: Given a set of $2n$ positive integers $X = \{x_1, x_2, \ldots, x_{2n}\}$, is there a partition of $X$ into two subsets $X_1$ and $X_2$ such that $\sum_{x_i \in X_1} x_i = \sum_{x_i \in X_2} x_i$ and such that, for each $i \in 1, 2, \ldots, n$, $X_1$ (and also $X_2$) contains exactly one element of $\{x_{2i-1}, x_{2i}\}$?

Let an instance of EOP be given by $2n$ numbers $\{x_1, x_2, \ldots, x_{2n-1}, x_{2n}\}$. Construct a path (see Figure 2) with set of nodes $\{s, b_1, b_2, \ldots, b_{2n}, z, z', c_{2n}, c_{2n-1}, \ldots, c_1, y, u\}$ ordered from left to right. The length of each edge corresponds to the Euclidean distance between its endpoints. We let $z' - z = b_{2i} - b_{2i-1} = 1$, $c_{2i-1} - c_{2i} = 1/2$, $z - b_{2n} = c_{2n} - z' = y - c_1 = L$, and $b_{2i+1} - b_{2i} = c_{2i} - c_{2i+1} = L$ for all $i \in [n-1]$ where $L = 3Cn/2 + 7/4 + 1$. Furthermore, we let $b_1 - s = S$, where $S > 0$ is specified later. The node $u$ is placed so that $u - z = b_1 - s + Cn + 0.5$, where $C$ is a constant that we can set to $C = 3$. There are $4n + 3$ agents:

- Two agents $h_1, h_2$: These agents have speed 1. Agent $h_1$ is initially at node $s$ and can traverse the interval $[s, b_1]$. Agent $h_2$ is initially at node $u$ and can traverse the interval $[z, u]$.
- $2n$ agents $p_1, p_2, \ldots, p_{2n}$: These agents are initially at node $z$; each agent $p_i$ has speed $v_{p_i} = \frac{1}{C + x_i/M}$ where $M = \sum_{i \in [2n]} x_i$. Agents $p_{2i-1}$ and $p_{2i}$ for $i \in [n]$ can traverse the interval $[b_{2i-1}, c_{2i-1}]$.

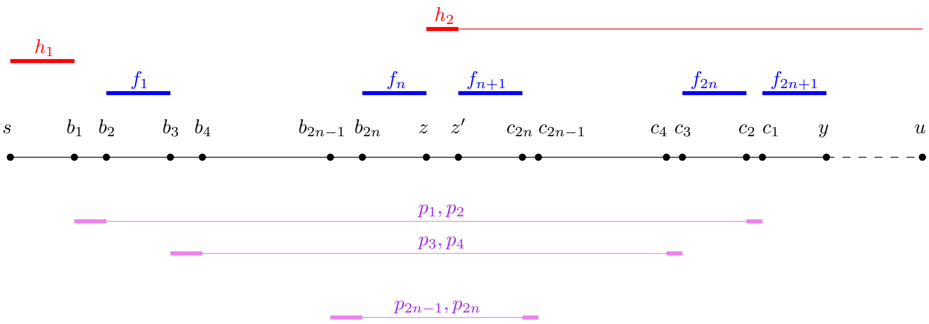


**Figure 2** Instance of $\mathsf{DDT_N}$ on a path.

- $2n+1$ agents $f_1, f_2, \ldots, f_{2n}, f_{2n+1}$: These agents have infinite speed. Each agent $f_i$ for $i < n$ is initially at node $b_{2i}$ and can traverse the interval $[b_{2i}, b_{2i+1}]$; agent $f_n$ is initially at node $b_{2n}$ and can traverse the interval $[b_{2n}, z]$; agent $f_{n+1}$ is initially at node $z'$ and can traverse the interval $[z', c_{2n}]$; each agent $f_i$ for $n+1 < i \le 2n$ is initially at node $c_{2(2n+1-i)+1}$ and can traverse the interval $[c_{2(2n+1-i)+1}, c_{2(2n+1-i)}]$; agent $f_{2n+1}$ is initially at node $c_1$ and can traverse the interval $[c_1, y]$.

The goal is to deliver a package from $s$ to $y$ as quickly as possible. We claim that there is a schedule with delivery time at most $T = S + 3Cn/2 + 7/4$ if and only if the original instance of EOP is a yes-instance.

First, assume that the instance of EOP is a yes-instance with partition $(X_1, X_2)$. Let $(P_1, P_2)$ be the corresponding partition of the set containing the $2n$ agents $p_i$. We construct a delivery schedule for the package as follows (the colors we mention refer to those shown in Figure 2). Until time $S = b_1 - s$, the agent $h_1$ carries the package to $b_1$, and all agents $p_i$ pick a bold purple interval and arrive at its left endpoint. This is done in such a way that agent $p_i$ picks its left bold purple interval (the interval of length 1 at the left end of its range) if $p_i \in P_1$, and its right bold purple interval (the interval of length $\frac{1}{2}$ at the right end of its range) otherwise. To guarantee that any agent $p_i$ arrives at the left endpoint of its interval by time $S$, we set $S = \max_{i\in[n]} \frac{(n+1-i)(L+1)}{\min\{p_{2i-1}, p_{2i}\}}$. Next, the package travels from $b_1$ to $z$ by always being alternatingly carried by an agent $p_i$ over a bold purple interval of length 1 and an infinite speed agent $f$ over a blue interval of length $L$. The package thus reaches $z$ at time $S + Cn + 1/2$ (because $\sum_{x_i \in X_1} x_i/M = 1/2$). As $u - z = S + Cn + 1/2$, the agent $h_2$ arrives at $z$ at exactly the same time. Then, the agent $h_2$ carries the package from $z$ to $z'$ in time 1. After that, the package is alternatingly carried by an infinite speed agent and an agent $p_i$ until it reaches $y$, taking time $Cn/2 + 1/4$. The total delivery time is $S + Cn + 1/2 + 1 + Cn/2 + 1/4 = S + 3Cn/2 + 7/4 = T$ as required.

Now consider the case that the original EOP instance is a no-instance. Assume that there exists a delivery schedule with delivery time $T = S + 3Cn/2 + 7/4$. First, we claim that neither an agent $p_i$ nor the agent $h_2$ can carry the package over an interval of length $L$ (these are the intervals $[b_{2n}, z]$, $[z', c_{2n}]$, $[c_1, y]$, and the intervals $[b_{2i}, b_{2i+1}]$, $[c_{2i+1}, c_{2i}]$ for any $i \in [n-1]$). Otherwise, the delivery time is larger than $S + L > S + 3Cn/2 + 7/4$ because these agents have speed at most 1 and these blue intervals have length $L = 3Cn/2 + 7/4 + 1$. Therefore, it is clear that each of the $2n$ agents $p_i$ and the agent $h_2$ will carry the package over exactly one of the following $2n+1$ intervals: the interval $[b_1, b_2]$, and the $2n$ intervals that lie between two consecutive blue intervals. Next, we claim that $h_2$ must carry the package over the interval $[z, z']$ with length 1. Otherwise, $h_2$ would have to carry the package over an interval $[c_{2i}, c_{2i-1}]$ with length $1/2$, and an agent $p_i$ with speed less than $1/C$ would have to carry the package through $[z, z']$ with length 1 instead. Even if we assume that all agents $p_i$ have the faster speed $1/C$, the resulting schedule would have delivery time at least $S + nC + C + (n-1)C/2 + 1/2 = S + 3Cn/2 + C/2 + 1/2$, which is larger than $T = S + 3Cn/2 + 7/4$ as $C = 3 > 5/2$.

This implies that agent $h_2$ carries the package over $[z, z']$ and, for each $i \in [n]$, one agent among $\{p_{2i-1}, p_{2i}\}$ carries the package on a bold purple interval to the left of $z$ and the other agent carries the package on a bold purple interval to the right of $z'$. Let the resulting partition of the agents $p_i$ be $(P_1, P_2)$, and let the corresponding partition of the original EOP instance be $(X_1, X_2)$. Suppose the package arrives at node $z$ at time $S + W$. Since the instance of EOP is a no-instance, either $W > Cn + 1/2$ or $W < Cn + 1/2$ holds. As the agents $p_i$ that carry the package to the left of $z$ take time $W$ in total, the agents $p_i$ that carry the package over intervals to the right of $z'$ take time $\frac{2Cn+1-W}{2}$ in total. Consider the two cases:

- $W > Cn + 1/2$. As agent $h_2$ carries the package from $z$ to $z'$, the delivery time is $S + W + 1 + \frac{2Cn+1-W}{2} = S + Cn + W/2 + 3/2 > S + 3Cn/2 + 7/4 = T$, a contradiction.
- $W < Cn + 1/2$. As agent $h_2$ carries the package from $z$ to $z'$, the package must wait at $z$ until time $S + Cn + 1/2$ when $h_2$ reaches $z$. The delivery time is $S + Cn + 1/2 + 1 + \frac{2Cn+1-W}{2} = S + 2Cn + 2 - W/2 > S + 2Cn + 2 - (Cn + 1/2)/2 = S + 3Cn/2 + 7/4 = T$, a contradiction.

Finally, we observe that in the constructed instance of DDT, the availability of edge handovers has no impact on the existence of a schedule with delivery time $S + 3Cn/2 + 7/4$. Therefore, the reduction establishes *NP*-hardness of both $\mathsf{DDT_N}$ and $\mathsf{DDT_E}$. ◀

## 6.2 Algorithms for drone delivery on a tree

Now, we show that all variants of the drone delivery problem can be solved optimally in polynomial time if the graph is a tree and all agents have the same speed (for minimizing the delivery time) or the same energy consumption rate (for minimizing the total energy consumption). In fact, the algorithms extend to the case of general graphs if the subgraph $G_a = (V_a, E_a)$ of each agent is *isometric*, i.e., it satisfies the following condition: For any two nodes $u, v \in V_a$, the length of the shortest $u$-$v$ path in $G_a$ is equal to the length of the shortest $u$-$v$ path in $G$. If the given graph is a tree, the subgraph $G_a$ of every agent $a$ is necessarily isometric because we assume that $G_a$ is connected and there is a unique path between any two nodes in a tree.

If all agents have the same speed (for DDT) or the same consumption rate (for DDC), handovers at internal points of edges can never improve the objective value. Therefore, we only need to consider $\mathsf{DDT_N}$ and $\mathsf{DDC_N}$ in the following. The crucial ingredient of our algorithms for the case of isometric subgraphs $G_a$ is:

▶ **Lemma 12.** *Consider a delivery schedule $\mathcal{S}$ that may use an arbitrary number of copies of each agent. If the subgraph of each agent is isometric and all agents have the same speed (or the same energy consumption rate), then $\mathcal{S}$ can be transformed in time $O(k(m + n \log n))$ into a schedule in which each agent is used at most once without increasing the delivery time (or the energy consumption).*

**Proof.** Consider the first agent $a$ that is used at least twice in $\mathcal{S}$. Assume that it carries the package from $u_{i-1}$ to $u_i$ in its first trip and from $u_{j-1}$ to $u_j$ in its last trip. Change the schedule so that $a$ carries the package from $u_{i-1}$ to $u_j$, and discard the trips by agents $i+1, \ldots, j-1$. As $G_a$ is isometric, the agent $a$ carries the package from $u_{i-1}$ to $u_j$ along a shortest path in $G$, and hence neither the time (in case of equal speed) nor the energy consumption (in case of equal energy consumption rate) increase by this modification. Repeat the modification step until every agent is used at most once.

There are at most $k-1$ modification steps, and each of them can be implemented in $O(m + n \log n)$ time using Dijkstra's algorithm with Fibonacci heaps [10]. ◀

▶ **Theorem 13.** $\mathsf{DDT_N}$ *(and* $\mathsf{DDT_E}$*) can be solved optimally in time* $O(k(n \log n + m))$ *if all agents have the same speed and the subgraph of every agent is isometric.*

**Proof.** Compute an optimal delivery schedule that may use multiple copies of each agent using Lemma 6 and then apply Lemma 12. ◀

▶ **Theorem 14.** $\mathsf{DDC_N}$ *(and* $\mathsf{DDC_E}$*) can be solved optimally in time* $O(nk^2 + n^2k)$ *if all agents have the same speed and the subgraph of every agent is isometric.*

**Proof.** Compute an optimal delivery schedule that may use multiple copies of each agent using Lemma 9 and then apply Lemma 12. ◀

The problem variants without initial positions can also be solved optimally in polynomial time: We simply compute a shortest $s$-$y$-path $P$ and place on each edge $e$ of $P$ a copy of an arbitrary agent in $A(e)$ and then apply Lemma 12.

For the special case where $G$ is a tree, the running-time for $\mathsf{DDT_N}$ can be improved to $O(kn)$ by using a simple algorithm that can even be implemented in a distributed way: The package acts as a magnet, and each agent moves towards the package until it meets the package and then follows it (or carries it) towards $y$ as long as its range allows. When several agents are at the same location as the package, the one whose range extends furthest towards $y$ carries the package. We refer to the full version [11] for details.

## 7 Conclusions

In this paper we have studied drone delivery problems in a setting where the movement area of each drone is restricted to a subgraph of the whole graph. For DDT, we have presented a strong inapproximability result and given a matching approximation algorithm. For DDC, we have shown *NP*-hardness and presented a 2-approximation algorithm. For the interesting special case of a path, we have shown that DDT is *NP*-hard if the agents can have different speeds. For trees (or, more generally, the case where the subgraph of each agent is isometric), we have shown that all problem variants can be solved optimally in polynomial time if the agents have the same speed or the same energy consumption.

We leave open the complexity of DDC on a path. For the case without initial positions, the complexity of both DDC and DDT on a path remains open. For DDT with initial positions on a path, a very interesting question is how well the problem can be approximated.

## References


1 Amazon Staff. Amazon Prime Air prepares for drone deliveries. `https://www.aboutamazon.com/news/transportation/amazon-prime-air-prepares-for-drone-deliveries`, 13 june 2022. Accessed: 2022-06-19.

2 Andreas Bärtschi, Jérémie Chalopin, Shantanu Das, Yann Disser, Barbara Geissmann, Daniel Graf, Arnaud Labourel, and Matús Mihalák. Collaborative delivery with energy-constrained mobile robots. *Theor. Comput. Sci.*, 810:2–14, 2020. doi:10.1016/j.tcs.2017.04.018.

3 Andreas Bärtschi, Jérémie Chalopin, Shantanu Das, Yann Disser, Daniel Graf, Jan Hackfeld, and Paolo Penna. Energy-efficient delivery by heterogeneous mobile agents. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 10:1–10:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.10.

4 Andreas Bärtschi, Daniel Graf, and Matús Mihalák. Collective fast delivery by energy-efficient agents. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018*, volume 117 of *LIPIcs*, pages 56:1–56:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.56.

5 Andreas Bärtschi and Thomas Tschager. Energy-efficient fast delivery by mobile agents. In Ralf Klasing and Marc Zeitoun, editors, *Fundamentals of Computation Theory - 21st International Symposium, FCT 2017, Bordeaux, France, September 11-13, 2017, Proceedings*, volume 10472 of *Lecture Notes in Computer Science*, pages 82–95. Springer, 2017. doi:10.1007/978-3-662-55751-8_8.

6 Iago A. Carvalho, Thomas Erlebach, and Kleitos Papadopoulos. On the fast delivery problem with one or two packages. *J. Comput. Syst. Sci.*, 115:246–263, 2021. doi:10.1016/j.jcss.2020.09.002.

7 Jérémie Chalopin, Shantanu Das, Yann Disser, Arnaud Labourel, and Matús Mihalák. Collaborative delivery on a fixed path with homogeneous energy-constrained agents. *Theor. Comput. Sci.*, 868:87–96, 2021. doi:10.1016/j.tcs.2021.04.004.

8 Jérémie Chalopin, Shantanu Das, Matúš Mihalák, Paolo Penna, and Peter Widmayer. Data delivery by energy-constrained mobile agents. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS 2013)*, volume 8243 of *Lecture Notes in Computer Science*, pages 111–122. Springer, 2013. doi:10.1007/978-3-642-45346-5_9.

9 Jérémie Chalopin, Riko Jacob, Matúš Mihalák, and Peter Widmayer. Data delivery by energy-constrained mobile agents on a line. In *41st International Colloquium on Automata, Languages, and Programming (ICALP 2014)*, volume 8573 of *Lecture Notes in Computer Science*, pages 423–434. Springer, 2014. doi:10.1007/978-3-662-43951-7_36.

10 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 4th Edition.* MIT Press, 2022. URL: https://mitpress.mit.edu/books/introduction-algorithms-fourth-edition.

11 Thomas Erlebach, Kelin Luo, and Frits C.R. Spieksma. Package delivery using drones with restricted movement areas. *CoRR*, abs/2209.12314, 2022. doi:10.48550/arXiv.2209.12314.

12 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.