Collaborative Delivery with Energy-Constrained Mobile Robots[☆]

Andreas Bärtschi[a,*], Jérémie Chalopin[b], Shantanu Das[b], Yann Disser[c], Barbara Geissmann[a], Daniel Graf[a], Arnaud Labourel[b], Matúš Mihalák[d]

[a] *Department of Computer Science, ETH Zürich, Switzerland*
[b] *LIF, CNRS et Aix-Marseille Université, France*
[c] *Institut für Mathematik, Technische Universität Darmstadt, Germany*
[d] *Department of Data Science and Knowledge Engineering, Maastricht University, Netherlands*

Abstract

We consider the problem of collectively delivering some package from a specified source to a designated target location in a graph, using multiple mobile agents. Each agent has a limited energy which constrains the distance it can move. Hence multiple agents need to collaborate to move the package, each agent handing over the package to the next agent to carry it forward. Given the positions of the agents in the graph and their respective budgets, the problem of finding a feasible movement schedule for the agents can be challenging. We consider two variants of the problem: in *non-returning* delivery, the agents can stop anywhere; whereas in *returning* delivery, each agent needs to return to its starting location, a variant which has not been studied before.

We first provide a polynomial-time algorithm for returning delivery on trees, which is in contrast to the known (weak) NP-hardness of the non-returning version. In addition, we give resource-augmented algorithms for returning delivery in general graphs. Finally, we give tight lower bounds on the required resource augmentation for both variants of the problem. In this sense, our results close the gap left by previous research.

Keywords: delivery; mobile agents; limited battery; resource augmentation; budget

1. Introduction

We consider a team of mobile robots which are assigned a task that they need to perform collaboratively. Even simple tasks such as collecting an item

[☆]This work was partially supported by the ANR project ANCOR (anr-14-CE36-0002-01), and by the SNF (project 200021L_156620).

[*]Corresponding author
Email address: baertschi@inf.ethz.ch (Andreas Bärtschi)

and delivering it to a target location can become challenging when it involves the cooperation of several agents. The difficulty of collaboration can be due to several limitations of the agents, such as limited communication, restricted vision or the lack of persistent memory, and this has been the subject of extensive research (see [1] for a recent survey). When considering agents that move physically (such as mobile robots or automated vehicles), a major limitation of the agents are their energy resources, which restricts the travel distance of the agent. This is particularly true for small battery operated robots or drones, for which the energy limitation is the real bottleneck. We consider a set of mobile agents where each agent i has a budget B_i on the distance it can move, as in [2, 3]. We model their environment as an undirected edge-weighted graph G, with each agent starting on some vertex of G and traveling along edges of G, until it runs out of energy and stops forever. In this model, the agents are obliged to collaborate as no single agent can usually perform the required task on its own. Our goal is to design centralized algorithms for the agents for such a type of collaboration.

The problem we consider is that of moving some physical item (henceforth called *package*) from a given source location to a target location in the graph G using a subset of the agents. Although the problem sounds simple, finding a valid schedule for the agents to deliver the package is computationally hard, even if we are given full information on the graph and the location of the agents. Given a graph G with designated source and target vertices, and k agents with given starting locations and energy budgets, the decision problem of whether the agents can collectively deliver a single package from the source to the target node in G is called BudgetedDelivery. Chalopin et al. [3, 4] showed that *Non-Returning* BudgetedDelivery is weakly NP-hard on paths and strongly NP-hard on general graphs.

Unlike previous papers, we also consider a version of the problem where each agent needs to return to its starting location after completing its task. This is a natural assumption, e.g. for robots that need to return to their docking station for maintenance or recharging. We call this variant *Returning* BudgetedDelivery. Surprisingly, this variant of the problem is easier to solve when the graph is a tree (unlike the original version of the problem), but we show it to be strongly NP-hard even for planar graphs. We present a polynomial time algorithm for solving *Returning* BudgetedDelivery on trees.

For arbitrary graphs, we are interested in resource-augmented algorithms. Since finding a feasible schedule for BudgetedDelivery is computationally hard when the agents have just enough energy to make delivery possible, we consider augmenting the energy of each robot by a constant factor γ, to enable a polynomial-time solution to the problem. Given an instance of BudgetedDelivery and some $\gamma > 1$, we have a γ-resource-augmented algorithm, if the algorithm, running in polynomial time, either (correctly) answers that there is no feasible schedule, or finds a feasible schedule for the modified instance with augmented budgets $\hat{B}_i = \gamma \cdot B_i$ for each agent i.

Our Model. We consider an undirected edge-weighted graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges. The weight (think length) $w(e)$ of an edge $e \in E$ defines the energy required to cross the edge in either direction in the following sense: We have k mobile agents which are initially placed on arbitrary nodes $p_1, \ldots, p_k$ of G, called starting positions. Each agent i has an initially assigned energy budget $B_i \in \mathbb{R}_{\geq 0}$ and can move along the edges of the graph, for a total distance of at most B_i (if an agent travels only on a part of an edge, its traveled distance is downscaled proportionally to the part traveled). The agents are required to move a single package from a given source node s to a target node t. An agent can pick up the package from its current location, carry it to another location (a vertex or a point inside an edge), and drop it there. Agents have global knowledge of the graph and are controlled by a central entity.

Given a graph G with vertices $s \neq t \in V(G)$ and the starting nodes and budgets for the k agents, we define BudgetedDelivery as the decision problem of whether the agents can collectively deliver the package without exceeding their individual budgets. In *Returning* BudgetedDelivery each agent needs to return to its respective starting position before using up its energy budget; in the *Non-Returning* version we do not place such a restriction on the agents and an agent may terminate at any location in the graph.
A solution to BudgetedDelivery is given in the form of a *schedule* which prescribes for each agent whether it moves and if so, the two locations in which it has to pick up and drop off the package. A schedule is *feasible* if the package can be delivered from s to t.

Related Work. Delivery problems in the graph have been usually studied for a single agent moving in the graph. For example, the well known *Traveling salesman problem* (TSP) or the *Chinese postman problem* (CPP) require an agent to deliver packets to multiple destinations located in the nodes of the graph or the edges of the graph. The optimization problem of minimizing the total distance traveled is known to be NP-hard [5] for TSP, but can be solved in polynomial time for the CPP [6].

When the graph is not known in advance, the problem of exploring a graph by a single agent has been studied with the objective of minimizing the number of edges traversed (see e.g. [7, 8]). Exploration by a team of two agents that can communicate at a distance has been studied by Bender and Slonim [9] for digraphs without node identifiers. The model of energy-constrained robot was introduced by Betke et al. [10] for single agent exploration of grid graphs. Later Awerbuch et al. [11] studied the same problem for general graphs. In both these papers, the agent could return to its starting node to refuel and between two visits to the starting node, the agent could traverse at most B edges. Duncan et al. [12] studied a similar model where the agent is tied with a rope of length B to the starting location and they optimized the exploration time, giving an $\mathcal{O}(m)$ time algorithm.

For energy-constrained agents without the option of refueling, multiple agents may be needed to explore even graphs of restricted diameter. Given a graph G and k agents starting from the same location, each having an energy con-

straint of B, deciding whether G can be explored by the agents is NP-hard, even if the graph G is a tree [13]. Dynia et al. studied the online version of the problem [14, 15]. They presented algorithms for exploration of trees by k agents when the energy of each agent is augmented by a constant factor over the minimum energy B required per agent in the offline solution. Das et al. [16] presented online algorithms that optimize the number of agents used for tree exploration when each agent has a fixed energy bound B. On the other hand, Dereniowski et al. [17] gave an optimal time algorithm for exploring general graphs using a large number of agents. Ortolf et al. [18] showed bounds on the competitive ratio of online exploration of grid graphs with obstacles, using k agents.

When multiple agents start from arbitrary locations in a graph, optimizing the total energy consumption of the agents is computationally hard for several formation problems which require the agents to place themselves in desired configurations (e.g. connected or independent configurations) in a graph. Demaine et al. [19] studied such optimization problems for multiple identical agents and provided approximation algorithms and inapproximability results. Similar problems have been studied for agents moving in the visibility graphs of simple polygons; optimizing either the total energy consumed or the maximum energy consumed per agent can be hard to approximate even in this setting, as shown by Bilo et al. [20].

A recent paper studies energy-efficient delivery of multiple packages by heterogeneous agents, which have different rates of energy consumption (but no constraints on the amount of energy they can use) [21, 22].

Anaya et al. [2] studied centralized and distributed algorithms for the information exchange by energy-constrained agents, in particular the problem of transferring information from one agent to all others (*Broadcast*) and from all agents to one agent (*Convergecast*). For both problems, they provided hardness results for trees and approximation algorithms for arbitrary graphs. The budgeted delivery problem was studied by Chalopin et al. [3] who presented hardness results for general graphs as well as resource-augmented algorithms. For the simpler case of lines, [4] proved that the problem is weakly NP-hard and presented a quasi-pseudo-polynomial time algorithm. Czyzowicz et al. [23] recently showed that the problems of budgeted delivery, broadcast and convergecast remain NP-hard for general graphs even if the agents are allowed to exchange energy when they meet.

Our Contribution. This is the first paper to study the *Returning* version of BudgetedDelivery. We first show that this problem can be solved in $\mathcal{O}(n + k \log k)$ time for lines and trees (Section 2). This is in sharp contrast to the *Non-Returning* version which was shown to be weakly NP-hard [4] even on lines. In Section 4, we prove that *Returning* BudgetedDelivery is NP-hard even for planar graphs. For arbitrary graphs with arbitrary values of agent budgets, we present a 2-resource-augmented algorithm and we prove that this is the best possible, as there exists no $(2-\epsilon)$-resource-augmented algorithm unless P = NP (Section 5). We show that this bound can be broken when the agents have the

same energy budget and we present a $(2 - 2/k)$-resource-augmented algorithm for this case.

For the *Non-Returning* version of the BudgetedDelivery, we close the gaps left open by previous research [3, 4]. In particular we prove that this variant of the problem is also strongly NP-hard on planar graphs, while it was known to be strongly NP-hard for general graphs and weakly NP-hard on trees. We also show tightness of the 3-resource-augmented algorithm for the problem, presented in [3]. Finally, in Section 6, we investigate the source of hardness for BudgetedDelivery and show that the problem becomes easy when the order in which the agents pick up the package is known in advance.

2. Returning BudgetedDelivery on the Tree

We study the *Returning* BudgetedDelivery on a tree and show that it can be solved in polynomial time. We immediately observe that this problem is reducible to the *Returning* BudgetedDelivery on a path: There is a unique s-t path on a tree and we can move each agent from her starting position to the nearest node on this s-t path while subtracting from her budget twice the distance traveled. The path problem now has an equivalent geometric representation on the line: the source node s, the target node t, and the starting positions of the agents p_i are coordinates of the real line. We assume $s < t$, i.e., the package needs to be delivered from left to right.

Without loss of generality, we consider schedules in which every agent i that moves uses all its budget B_i. Because every agent needs to return to its starting position, an agent i can carry the package on any interval of size $B_i/2$ that contains the starting position p_i. For every agent i, let $l_i = p_i - B_i/2$ denote the leftmost point where she can pick up a package, and let $r_i = p_i + B_i/2$ be the rightmost point to where she can deliver the package. The *Returning* BudgetedDelivery on a line now becomes the following *covering problem*: Can we choose, for every i, an interval I_i of size $B_i/2$ that lies completely within the *region* $[l_i, r_i]$ such that the segment $[s, t]$ is covered by the chosen intervals, i.e., such that $[s, t] \subseteq \bigcup_i I_i$?

The following *greedy* algorithm solves the covering problem. The algorithm works iteratively in rounds $r = 1, 2, \ldots$. We initially set $s_1 = s$. We stop the algorithm whenever $s_r \geq t$, and return true. In round r, we pick i^* having the smallest r_{i^*} among all not yet used agents i with $l_i \leq s_r < r_i$, and set $s_{r+1} = \min\{r_{i^*}, s_r + B_{i^*}/2\}$ and $I_{i^*} = (s_{r+1} - B_{i^*}/2, s_{r+1})$, and continue with the next round $r + 1$. If we cannot choose i^*, we stop the algorithm and return false.

Theorem 1. *There is an $\mathcal{O}(n + k \log k)$-time algorithm for* Returning BudgetedDelivery *on a tree.*

Proof. The reduction from a tree to a path takes $\mathcal{O}(n)$ time using breadth-first search from s and the algorithm *greedy* can be implemented in time $\mathcal{O}(k \log k)$ using a priority queue. For the correctness, we now show that greedy returns a solution to the covering problem if and only if there exists one.

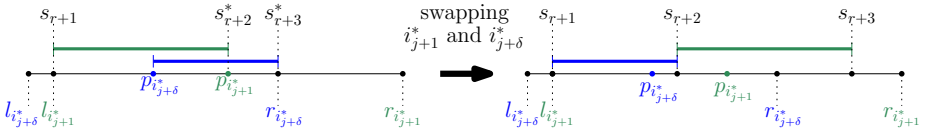

Figure 1: Changing the order of agents i^*_{j+1} and $i^*_{j+\delta}$ in the schedule.

Greedy can be seen as advancing the cover of $[s,t]$ from left to right by adding intervals I_i. Whenever it decides upon I_i, it will set s_r to the respective endpoint of I_i, and never ever consider i again or change the placement of I_i within the boundaries $[l_i, r_i]$. Thus, whenever $s_r \geq t$, the intervals I_i form a cover of $[s,t]$.

We now show that if a cover exists, greedy finds one. Observe first that a cover can be given by a subset of the agents $\{i_1, \ldots, i_t\}$, $t \leq k$, and by their ordering $(i_1, i_2, \ldots)$, according to the right endpoints of their intervals I_{i_j}, since we can reconstruct a covering by always placing the respective interval I_{i_j} at the rightmost possible position.

Suppose, for contradiction, that greedy fails. Let $(i^*_1, i^*_2, \ldots)$ be a minimal cover of $[s,t]$ that agrees with the greedy schedule $(i_1, i_2, \ldots)$ in the maximum number of first agents $i_1, \ldots, i_j$. Hence, $j+1$ is the first position such that $i^*_{j+1} \neq i_{j+1}$. The left endpoints of I^*_{j+1} and I_{j+1} correspond to s_{r+1} in our algorithm. If agent i_{j+1} does not appear in the solution $(i^*_1, i^*_2, \ldots)$, adding i_{j+1} to that solution and deleting (if the solution is not a minimal cover) some of the subsequent ones results in a minimal cover that agrees on the first $j+1$ agents, a contradiction.

If agent i_{j+1} appears in the solution $(i^*_1, i^*_2, \ldots)$, say, as agent $i^*_{j+\delta}$, for some $\delta \geq 2$, then we modify this cover by swapping i^*_{j+1} with $i^*_{j+\delta}$. We claim that the new solution still covers $[s,t]$. To see this, let us first show that $\delta = 2$. Recall that both the agent $i_{j+1} = i^*_{j+\delta}$ and the agent i^*_{j+1} can extend the covering beyond s_{r+1}. By the greedy choice, it follows that $r_{i_{j+1}} = r_{i^*_{j+\delta}} \leq r_{i^*_{j+1}}$. Since every agent i covers with its interval I_i half of its region $[l_i, r_i]$, it follows that in the solution $(i^*_1, i^*_2, \ldots)$ agents i^*_{j+1} and $i^*_{j+\delta}$ together cover the region $[s_{r+1}, r_{i^*_{j+\delta}}]$ (both agents can place an interval at s_{r+1}, and both agents' intervals have length at least half of the length of the region $[s_{r+1}, r_{i^*_{j+\delta}}]$). By the minimality of the solution $(i^*_1, i^*_2, \ldots)$, it follows that $i^*_{j+\delta} = i^*_{j+2}$. Obviously, the two agents do not cover more than this region (because agent $i^*_{j+\delta}$'s righmost endpoint of its interval is $r_{i^*_{j+\delta}}$). Now, since $\delta = 2$, and because of the fact that agents i^*_{j+1} and $i^*_{j+\delta}$ can place their intervals within the region $[s_{r+1}, r_{i^*_{j+\delta}}]$, it follows that exchanging the order of the two agents produces a solution where the region $[s_{r+1}, r_{i^*_{j+\delta}}]$ will be still covered (and perhaps even more), and the agents $i^*_{j+3}, i^*_{j+4}, \ldots$ can place the intervals in the very same way. See Figure 1 for illustration of the situation. □

3. Resource Augmentation Algorithms

We now look at general graphs $G = (V, E)$. As we will see in the next section, BUDGETEDDELIVERY is NP-hard, hence we augment the budget of each agent by a factor $\gamma > 1$ to allow for polynomial-time solutions. For non-returning agents, a $\min\left\{3, 1 + \max \frac{B_i}{B_j}\right\}$-resource-augmented algorithm was given by Chalopin et. al. [3]. We first provide a 2-resource-augmented algorithm for *Returning* BUDGETEDDELIVERY. This is tight as there is no polynomial-time $(2-\varepsilon)$-resource-augmented algorithm, unless P = NP (Section 5). If, however, the budgets of the agents are similar, we can go below the 2-barrier: In this case, we present a $\left(1 + \frac{k-2}{k} \max \frac{B_i}{B_j}\right)$-resource-augmented algorithm. Throughout this section, we assume that there is no feasible schedule with a single agent, which we can easily verify.

Preliminaries. We denote by $d(u, v)$ the distance of two points $u, v \in G$. Assume an agent i with budget B_i starts in u and moves first to v. Which locations in the graph (vertices and positions on the edges) are still reachable by i so that he has sufficient energy left to move back to u? We define the ellipsoid $\mathcal{E}(u, v, B_i) = \{p \in G \mid d(u, v) + d(v, p) + d(p, u) \leq B_i\}$ and the ball $\mathcal{B}(u, \frac{B_i}{2}) = \mathcal{E}(u, u, B_i)$. It is easy to see that $\mathcal{E}(u, v, B_i)$ can be (i) computed in polynomial time by running Dijkstra's shortest path algorithm from both u and v and (ii) represented in linear space: We store all vertices $p \in V$ with $p \in \mathcal{E}(u, v, B_i)$, and for each edge $(p, q) \in E$ with $p \in \mathcal{E}(u, v, B_i), q \notin \mathcal{E}(u, v, B_i)$ we store the furthest point of (p, q) still reachable by i.

Theorem 2 (2-resource-augmentation)**.** *There is a polynomial-time* 2*-resource-augmented algorithm for* Returning BUDGETEDDELIVERY*.*

Proof. Denote by p_i the starting position of agent i. We consider the balls $\mathcal{B}_i := \mathcal{B}(p_i, \frac{B_i}{2})$ around all agents, as well as the balls $\mathcal{B}(s, 0)$ and $\mathcal{B}(t, 0)$ of radius 0 around s and t. We compute the *intersection graph* G_I of the balls, which can be done in polynomial time. If there is a feasible schedule, then there must be a path from $\mathcal{B}(s, 0)$ to $\mathcal{B}(t, 0)$ in G_I (for example the path given by the balls around the agents in the feasible schedule).

If there is no path from $\mathcal{B}(s, 0)$ to $\mathcal{B}(t, 0)$, then the algorithm outputs that there is no feasible schedule with non-augmented budgets. Otherwise we can get a 2-resource-augmentation as follows: Pick a shortest path from $\mathcal{B}(s, 0)$ to $\mathcal{B}(t, 0)$ in G_I and denote by $\ell \leq k$ the number of agents on this path, labeled without loss of generality $1, 2, \ldots, \ell$. For each edge on the shortest path, we specify a handover point $h_i \in \mathcal{B}_i \cap \mathcal{B}_{i+1}$ in G (where we set $h_0 = s$ and $h_\ell = t$). Then each agent i, $i = 1, \ldots, \ell$ walks from its starting position p_i to the handover point h_{i-1} to pick up the package, goes on to the handover point h_i to drop the package there, and returns home to p_i. Since $h_{i-1}, h_i \in \mathcal{B}(p_i, \frac{B_i}{2})$, the budget needed by agent i to do so is at most $\hat{B}_i = d(p_i, h_{i-1}) + d(h_{i-1}, h_i) + d(h_i, p_i) \leq \frac{B_i}{2} + 2 \cdot \frac{B_i}{2} + \frac{B_i}{2} = 2 \cdot B_i$. □

Theorem 3. *There is a polynomial-time* $\left(1 + \frac{k-2}{k} \max \frac{B_j}{B_i}\right)$*-resource-augmented algorithm for* Returning BUDGETEDDELIVERY*.*

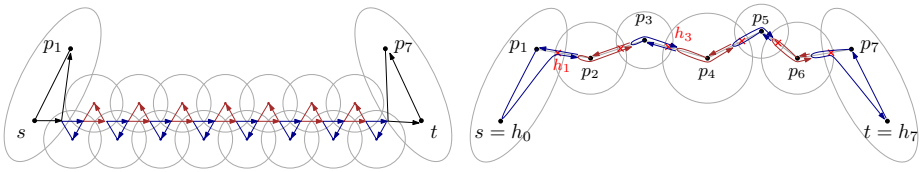

Figure 2: (left) Feasible schedule. (right) Schedule with $\left(1 + \frac{5}{7} \max \frac{B_j}{B_i}\right)$-resource-augmentation.

Proof. We first "guess" the first agent a and the last agent b of the feasible schedule (by trying all $\binom{k}{2}$ pairs). In contrast to Theorem 2, we can in this way get a 2-resource-augmented solution in which a and b only need their original budgets. Intuitively, we can evenly redistribute the remaining part of $\hat{B}_a$ and $\hat{B}_b$ among all k agents, such that for each agent i we have $\hat{B}_i \leq B_i + \frac{k-2}{k} \max B_j$. Without loss of generality, we assume that agent a walks from its starting position on a shortest path to s to pick up the package, and that agent b walks home directly after dropping the package at t. Hence consider the ellipsoids $\mathcal{B}_a := \mathcal{E}(p_a, s, B_a)$ and $\mathcal{B}_b := \mathcal{E}(p_b, t, B_b)$ as well as the balls $\mathcal{B}_i := \mathcal{B}(p_i, \frac{B_i}{2})$ around the starting positions of all other agents and compute their intersection graph G_I.

We denote by $i = 1, \ldots, \ell$ the agents on a shortest path from $\mathcal{B}_a$ to $\mathcal{B}_b$ in G_I (if any), where $a = 1$, $b = \ell \leq k$ and specify the following points: $h_0 = s$, $h_i \in \mathcal{B}_i \cap \mathcal{B}_{i+1}$, and $h_\ell = t$. If the agents handover the package at the locations h_i, we get a 2-resource-augmentation where the agents 1 and ℓ use only their original budget. Instead we let them help their neighbours 2 and $\ell - 1$ by $\frac{\ell-2}{\ell} B_2$ and $\frac{\ell-2}{\ell} B_{\ell-1}$, respectively. Those agents further propagate the surplus towards the agent(s) in the middle, see Figure 2 (right). Specifically, we let the agents move as follows:

Agent 1 goes to h_0 to pick up the package and then goes on to h_1. Then he moves towards p_2 along the shortest path from h_1 to p_2 by a $\frac{\ell-2}{\ell}$-fraction of $d(h_1, p_2)$, drops off the package and returns home. Since $d(h_1, p_2) \leq \frac{B_2}{2}$, the budget needed to do so is at most $d(p_1, h_0) + d(h_0, h_1) + 2\frac{\ell-2}{\ell} d(h_1, p_2) + d(h_1, p_1) \leq B_1 + \frac{\ell-2}{\ell} B_2$. Agents $i = 2, \ldots, \lfloor \frac{\ell}{2} \rfloor$ get help from their preceding agent and thus can help the following agent: Agent i walks from its starting position p_i by a $\frac{2(i-1)}{\ell}$-fraction towards h_{i-1} to pick up the package and then returns home. Then i goes on to the point h_i and from there on by a $\frac{\ell-2i}{\ell}$-fraction towards p_{i+1} to drop off the package. Finally, agent i returns home to p_i. Since $h_{i-1}, h_i \in \mathcal{B}_i$ and $h_i \in \mathcal{B}_{i+1}$, the budget needed by agent i to do so is at most $2\frac{2(i-1)}{\ell} d(p_i, h_{i-1}) + 2d(p_i, h_i) + 2\frac{\ell-2i}{\ell} d(h_i, p_{i+1}) \leq \frac{2(i-1)}{\ell} B_i + B_i + \frac{\ell-2i}{\ell} B_{i+1} \leq B_i + \frac{\ell-2}{\ell} \max\{B_i, B_{i+1}\}$. Agents $i = \lceil \frac{\ell+2}{2} \rceil, \ldots, \ell$ help in the same way their preceding agent, hence they need a budget of at most $B_i + \frac{\ell-2}{\ell} \max\{B_{i-1}, B_i\}$. If ℓ is odd there is an additional middle agent $i = \frac{\ell+1}{2}$ who needs a budget of at most $2\frac{\ell-1}{\ell} d(p_i, h_{i-1}) + 2\frac{\ell-1}{\ell} d(p_i, h_{i+1}) \leq 1 + \frac{\ell-2}{\ell} B_i$. Hence we achieve a resource augmentation of $1 + \frac{\ell-2}{\ell} \max \frac{B_j}{B_i} \leq 1 + \frac{k-2}{k} \max \frac{B_j}{B_i}$. □

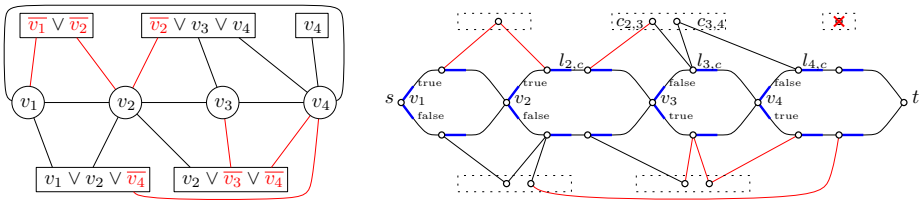

Figure 3: (left) A plane embedding of a 3CNF F which is satisfied by $(v_1, v_2, v_3, v_4) = (\text{true}, \text{false}, \text{false}, \text{true})$. (right) Its transformation to the corresponding delivery graph.

4. Hardness for Planar Graphs

In this section, we show that BUDGETEDDELIVERY in a planar graph is strongly NP-hard, both for the *Returning* version and the *Non-Returning* version. Both proofs are based on the same reduction from PLANAR3SAT.

Planar 3SAT. Let F be a conjunctive normal form 3CNF with a set of variables $V = \{v_1, \ldots, v_x\}$ and a set of clauses $C = \{c_1, \ldots, c_y\}$. Each clause is a disjunction of at most three literals $\ell(v_i) \vee \ell(v_j) \vee \ell(v_k)$, where $\ell(v_i) \in \{v_i, \overline{v_i}\}$. We can represent F by a graph $H(F) = (C \cup V, A_1 \cup A_2)$ which we build as follows: We start with a bipartite graph with the node set N consisting of all clauses and all variables and an edge set A_1 which contains an edge between each clause c and variable v if and only if v or $\overline{v}$ is contained in c, $A_1 = \{\{c_i, v_j\} \mid v_j \in c_i \text{ or } \overline{v_j} \in c_i\}$. To this graph we add a cycle A_2 consisting of edges between all pairs of consecutive variables, $A_2 = \{\{v_j, v_{j+1}\} \mid 1 \leq j < x\} \cup \{v_x, v_1\}$. We call F *planar* if there is a plane embedding of $H(F)$ which *at each variable node* has all paths representing positive literals on one side of the cycle A_2 and all paths representing negative literals on the other side of A_2. The decision problem PLANAR3SAT of finding whether a given planar 3CNF F is satisfiable or not is NP-complete, a result due to Lichtenstein [24]. We assume without loss of generality that every clause contains at most one literal per variable. For an example of such an embedding, see Figure 3 (left).

Building the Delivery Graph. We first describe how to turn a plane embedding of a planar 3CNF graph $H(F)$ into a delivery graph $G(F)$, see Figure 3. Only later we will define edge weights, the agents' starting positions and their energy budgets. We will focus on *Returning* BUDGETEDDELIVERY; the only difference for non-returning agents lie in their budgets, we provide adapted values for non-returning agents in footnotes.

We transform the graph in four sequential steps: First we dissolve the edge $\{v_x, v_1\}$ and replace it by an edge $\{v_x, v_{x+1}\}$. Secondly, denote by $\deg_{H(F),A_1}(v)$ the total number of positive literal edges and negative literal edges adjacent to v. Then we can "disconnect" and "reconnect" each variable node v_i $(1 \leq i \leq n)$ from all of its adjacent clause nodes as follows: We delete all edges $\{\{v_i, c\}\} \subseteq A_1$ and split $\{v_i, v_{i+1}\}$ into two paths $p_{i,\text{true}}$ and $p_{i,\text{false}}$, on which we place a total

of $\deg_{H(F),A_1}(v)$ internal *literal nodes* $l_{i,c}$: If v_i is contained in a clause c – and thus we previously deleted $\{v_i, c\}$ – we place $l_{i,c}$ on $p_{i,\text{false}}$ and "reconnect" the variable by adding an edge between $l_{i,c}$ and the clause node c. Else if $\overline{v_i}$ is contained in c we proceed similarly (putting the node $l_{i,c}$ on $p_{i,\text{true}}$ instead). As a third step, depending on the number of literals of each clause c, we may modify its node: If c contains only a single literal, we delete the c node (but we keep its literal node $l_{i,c}$). If c contains two literals $\ell(v_i), \ell(v_j)$, we rename the node to $c_{i,j}$. If c is a disjunction of three literals $\ell(v_i), \ell(v_j), \ell(v_k)$, we split it into two nodes $c_{i,j}$ (connected to $l_{i,c}$, $l_{j,c}$) and $c_{j,k}$ (connected to $l_{j,c}$, $l_{k,c}$). Finally, we place the package on the first variable node $s := v_1$ and set its destination to $t := v_{x+1}$.

We remark that all four steps can be implemented such that the resulting delivery graph $G(F)$ is still planar, as illustrated in Figure 3 (in each path tuple $(p_{i,\text{true}}, p_{i,\text{false}})$ the order of the internal nodes follows the original circular order of adjacent edges of v_i, and for each clause $c = \ell(v_i) \vee \ell(v_j) \vee \ell(v_k)$ the nodes $c_{i,j}$ and $c_{j,k}$ are placed close to each other).

Reduction Idea. We show that the package can't be delivered via any of the clause nodes. Thus the package has to be routed in each path pair $(p_{i,\text{true}}, p_{i,\text{false}})$ through exactly one of the two paths. If the package is routed via the path $p_{i,\text{true}}$, we interpret this as setting $v_i = \text{true}$ and hence we can read from the package trajectory a satisfiable assignment for F.

Agent Placement and Budgets. We will use Greek letters ζ, δ for weights that depend on each other or on the input. We place three kinds of agents on G:

1. *Variable agents:* x agents which are assigned to the variable nodes $v_1, \ldots, v_x$. These agents will have to decide whether the package is delivered via $p_{i,\text{true}}$ or via $p_{i,\text{false}}$, thus setting the corresponding variable to true or to false. We give all of them a budget of 2ζ.[1]
2. *Clause agents:* One agent per created clause *node*, e.g. a clause c containing three literals gets two agents, one in each of the two clause nodes. We think of these agents as follows: If in $c = \ell(v_i) \vee \ell(v_j) \vee \ell(v_k)$ the literal $\ell(v_j)$ is false, then clause c needs to send one of its agents down to the corresponding path node $l_{j,c}$ to help transporting the package over the adjacent "gap" of size ζ (depicted blue in Figures 3 (right), 4). A 3CNF F will be satisfiable, if and only if no clause needs to spend more agents than are actually assigned to it respectively its node(s) in $G(F)$. We give all clause agents a budget of $2 \cdot (1 + \zeta)$.[2]
3. *Separating agents:* These will be placed in-between the agents defined above, to ensure that the variable and clause agents actually need to solve the task intended for them (they should not be able to deviate and help

[1] In the *Non-Returning* version we want agents to have the same "range", hence we set their budget to ζ.

[2] In the *Non-Returning* version we assign a budget of $(1 + \zeta)$ to clause agents.

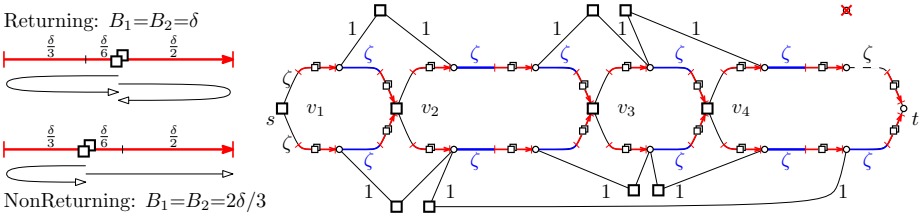

Figure 4: (left) Two examples of δ-tubes for both versions of BudgetedDelivery. (right) Agent placement and edge weights on $G(F)$; agents are depicted by squares

out somewhere else – not even their own kind). The separating agents will be placed in pairs inside δ-*tubes*, which we define below.

Remark 4. *Strictly speaking, a reduction without variable agents works as well. In terms of clarity, we like to think of variable agents as the ones setting the variables to* true *or* false*.*

δ-Tubes. We call a line segment a δ-*tube* if it satisfies the following four properties: (i) It has a length of δ. (ii) It contains exactly two agents which both have budget at most δ. (iii) Neither agent has enough energy to leave the line segment on the left or on the right by more than a distance of $\frac{\delta}{3}$. (iv) The agents can collectively transport a package through the line segment from left to right. δ-tubes exist for both BudgetedDelivery versions, examples are given in Figure 4 (left). The reader may think of these examples, whenever we talk about δ-tubes.

Edge Weights. We define edge weights on our graph $G(F)$ as follows: All edges between clause nodes and internal path nodes get weight 1 (in particular this means that if a clause agent walks to the path, it has a remaining range of ζ). Each path consists of alternating pieces of length ζ and of δ-tubes. We choose $\delta := \frac{4\zeta}{3} > \zeta$. This means that neither variable nor clause agents can cross a δ-tube (because their budget is not sufficiently large). Furthermore the distance any separating agent can move outside of its residential tube is at most $\frac{\delta}{3} = \frac{4\zeta}{9} < \frac{\zeta}{2}$. In particular separating agents are not able to collectively transport the package over a ζ-segment, since from both sides they are not able to reach the middle of the segment to handover the package. At last, $\zeta := \frac{1}{8}$.

Equivalence of planar 3SAT and planar BudgetedDelivery. In the following we show that solving BudgetedDelivery on $G(F)$ is at least as hard as solving the underlying planar 3SAT instance. We first establish three key properties (Propositions 1–3):

Proposition 1. *If a planar 3CNF F is satisfiable, then in the corresponding delivery graph $G(F)$, the agents can collectively deliver the package from s to t and return to their respective starting positions.*

Proof. Assume that there is a satisfiable assignment for F. Then the agents' actions are straightforward: Each *variable agent* placed on v_i moves according to the variable assignment to v_i by a ζ-distance into either the *true*-path $p_{i,\text{true}}$ or the *false*-path $p_{i,\text{false}}$. For the package to be delivered to the next variable agent, it needs to be handed across δ-tubes and ζ-segments. The former can always be done by the respective *separating agents* residing inside the δ-tube. It remains to be shown that the latter can be done by *clause agents*. To this end, consider a clause c which consists of $|c|$ literals.

If $|c| = 1$ respectively $c = \ell(v_j)$ for some j, then there is no clause node in $G(F)$ at all (see the top right clause in Figure 4). No agent can reach the ζ-segment adjacent to $l_{j,c}$, but this does not cause a problem, since by our assumption the literal $\ell(v_j)$ is satisfied and thus the variable agent at v_j chose to deliver the package via the opposite path $p_{j,\ell(v_j)}$.

If $|c| = 2$, then there is one clause agent on a single clause node $c_{i,j}$ which is connected to the internal path nodes $l_{i,c}$ and $l_{j,c}$ (see the top left clause in Figure 4). Both have adjacent ζ-segments which correspond to the literals $\overline{\ell(v_i)}, \overline{\ell(v_j)}$. By assumption, at least one literal – without loss of generality $\ell(v_j)$ – is satisfied, and since the variable agent choosing the assignment for v_j thus takes the "opposite" path $p_{j,\ell(v_j)}$, the ζ-segment corresponding to $\overline{\ell(v_j)}$ does not need to be crossed while delivering the package. If the other literal is not satisfied, then the clause agent is needed at the corresponding ζ-segment, otherwise it can stay at its place of origin.

If $|c| = 3$, we have three literals/ζ-segments and we have two clause nodes with one agent each (see the top center clause in Figure 4). One is connected to the first and the second ζ-segment, the other to the second and third. Collectively the two agents can reach every possible pair of segments out of the three ζ-segments. At least one literal $\ell(v_j)$ is satisfied. To each of the remaining ζ-segments we can therefore send one agent. Moving to the path needs 1 unit of energy (and so does returning to the clause node). Hence the agent can cover a remaining distance of ζ, which is sufficient to transport the package to the next δ-tube. □

Proposition 2. *It is not possible to deliver a package from s to t via any clause node of the delivery graph $G(F)$.*

Proof. For the sake of contradiction assume that the package is transported via a clause node $c_{i,j}$ which connects to the internal path nodes $l_{i,c}$ and $l_{j,c}$. Except for the clause agent stationed at $c_{i,j}$, no other agent can move further than $\zeta = \frac{1}{8}$ into each of the two edges $\{l_{i,c}, c_{i,j}\}$ and $\{l_{j,c}, c_{i,j}\}$. Hence the clause agent stationed at $c_{i,j}$ needs to cover in each edge a distance of $2(1-\zeta)$ (to go back and forth), hence for both edges it needs an energy of at least $2 \cdot 2(1-\zeta) = 4 \cdot \frac{7}{8} = \frac{7}{2}$. However, the agent has a budget of only $2(1+\zeta) = 2 \cdot \frac{9}{8} < \frac{7}{2}$, yielding a contradiction to the package being transported over the clause node.[3] □

[3]Non-returning agents need to cover only one of the edges twice (to go back and forth), hence they need an energy of at least $3(1-\zeta) = \frac{21}{8}$ versus a budget of $1+\zeta = \frac{9}{8}$, yielding a

Proposition 3. *To deliver the package over a ζ-segment adjacent to a variable node v_i, we need the* variable agent *with starting position v_i. To deliver the package over a ζ-segment adjacent to a literal node $l_{j,c}$, we need a* clause agent *with starting position $c_{i,j}$ or $c_{j,k}$.*

Proof. Recall that $\delta = \frac{4\zeta}{3}$. Separating agents inside δ-tubes can neither single-handedly nor collectively (starting from both sides) transport the package over a ζ-segment, since they can move outside of their residential tube by at most $\frac{\delta}{3} < \frac{\zeta}{2}$. Furthermore variable agents and clause agents can move on a true- or false-path inside an interval of size at most $\zeta < \delta$, hence they can't cross a δ-tube. Thus to transport the package over a ζ-segment adjacent to a variable node v_j, we need the variable agent placed on v_j. On the other hand, transporting the package over a ζ-segment adjacent to the internal path node $l_{j,c}$ needs a clause agent of clause c. If c has two clause nodes $c_{i,j}, c_{j,k}$, either of the two clause agents will do. □

Lemma 1 (Returning BudgetedDelivery)**.** *A planar 3CNF F is satisfiable if and only if it is possible to deliver a package from s to t in the corresponding delivery graph $G(F)$, such that all agents are still able to return to their starting points in the end.*

Proof. "⇒" This direction has been shown in Proposition 1.

"⇐" Assume that the package can be delivered from s to t. From Proposition 2 it follows that the package has to be transported through the true- and false-paths. Without loss of generality, the package must move monotonously through the paths $p_{i,\text{true}}$ or $p_{i,\text{false}}$. By Proposition 3 we know that for each ζ-segment that the package is delivered over, we need either the corresponding variable agent or the corresponding clause agent. It remains to show that we have enough clause agents for the task:

Each clause with $|c|$ literals "owns" only $|c| - 1$ clause agents and thus must have at least one satisfied literal (otherwise the $|c| - 1$ clause agents would not be sufficient to help in all corresponding ζ-segments). Hence we can read a satisfiable variable assignment for the Planar3SAT instance directly from the choice of the variable agents (which each pick the adjacent true- or the adjacent false-path). □

It is easy to see that the same arguments work for *Non-Returning* Budgeted-Delivery as well, hence we immediately get the same statement for the *Non-Returning* version.

Corollary 1 (Non-Returning BudgetedDelivery)**.** *A planar 3CNF F is satisfiable if and only if it is possible to deliver a package in the corresponding delivery graph.*

---

contradiction as well.

Recall that a delivery graph $G(F)$ created from a planar 3CNF F is planar. Furthermore the size of $G(F)$, as well as the number of agents we use, is polynomial in the number of clauses and variables. The agents' budgets and the edge weights are polynomial in ζ, δ and thus constant. Thus Lemma 1 shows NP-hardness of BudgetedDelivery on planar graphs. Finally, note that hardness also holds for a *uniform* budget B: One can simply add an edge of length $(B - B_i)/2$ to the starting location of each agent i and relocate i to the end of this edge.[4]

Theorem 5 (Hardness of BudgetedDelivery)**.** *Both versions of* BudgetedDelivery *are strongly* NP*-hard on planar graphs, even for uniform budgets.*

5. Hardness of Resource Augmentation

Main Ideas. We show that for all $\varepsilon > 0$, there is no polynomial-time $(2 - \varepsilon)$-resource-augmented algorithm for *Returning* BudgetedDelivery, unless P = NP. The same holds for $(3 - \varepsilon)$-resource-augmentation for the *Non-Returning* version. Intuitively, an algorithm which finds out how to deliver the package with resource-augmented agents will at the same time solve 3SAT. We start by taking the reduction from Planar3SAT from Section 4. However, in addition to the previous delivery graph construction $G(F)$, we need to replace the δ-tubes and ζ-segments in order to take care of three potential pitfalls. We illustrate the modification into the new graph $G'(F)$ in Figure 6:

1. In a resource-augmented setting, δ-tubes are no longer able to separate the clause and variable agents: These agents might be able to cross the δ-tube, or the separating agents residing inside the δ-tube can help out in the ζ-segments (there is no value for δ to prevent both). We will tackle this issue below by replacing δ-tubes by a chain of logarithmically many tubes with exponentially increasing and decreasing δ-values.
2. In the reduction for the original decision version of BudgetedDelivery, a clause c with three literals gave rise to two clause nodes $c_{i,j}$, $c_{j,k}$ that were adjacent to the same path node $l_{j,c}$. Hence the agent on $c_{i,j}$, now with resource-augmented budget, could pick up the package at $l_{j,c}$ and bring it close to the second resource-augmented agent stationed at $c_{j,k}$. This agent then might transport the package via its own clause node to the distant literal node $l_{k,c}$. To avoid this, we replace every ζ-segment adjacent to such a "doubly" reachable path node $l_{j,c}$ by two small parallel arcs. Both arcs contain exactly one ζ-segment, reachable from only one clause node (the package can then go over either arc), as well as a chain of tubes to provide the necessary separation.

[4] We relocate a non-returning agent by adding an edge of length $(B - B_i)$.

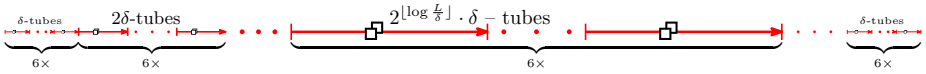

Figure 5: L-δ-chains consist of blocks of 6 tubes of exponentially increasing and decreasing size.

3. A single clause agent stationed at $c_{i,j}$ might retrieve the package from the first literal node $l_{i,c}$, walk back to its origin and then on to the second literal $l_{j,c}$, thus transporting the package over a clause node. This can always be done by 2-resource-augmented agents; however for $(2-\varepsilon)$-resource-augmentation we can prevent this by carefully tuning the weights of the ζ-segments, e.g. such that $(2-\varepsilon)\cdot(1+\zeta) \ll 2$.[5]

We now give a more formal description of the ideas mentioned above. Recall that a δ-tube had length δ and contained two agents with budget at most δ each. If these agents are now γ-resource-augmented, $\gamma < 3$, they can move strictly less than 3δ to the right or to the left of the δ-tube. In the following we want to uncouple the length of the line segment from the range the agents have left to move on the outside of the line segment.

L-δ-Chains. We call a line segment an *L-δ-chain* if it satisfies the following three properties: (i) Its length is at least L (a constant). (ii) No γ-resource-augmented agent ($1 \le \gamma < 3$) contained in the chain has enough energy to leave the line segment by 3δ or more. (iii) The agents contained in the chain can collectively transport a package through the line segment from left to right (already with their original budget).

We can create L-δ-chains for both BudgetedDelivery versions simply by using the respective *δ-tubes* as a black box: We start our line segment by adding a block of six δ-tubes next to each other, followed by a block of six 2δ-tubes, a block of six 4δ-tubes and so on until we get a block of length at least $6 \cdot 2^{\lfloor \log L/\delta \rfloor} \cdot \delta > L$. The same way we continue to add blocks of six tubes with lengths decreasing by powers of 2, see Figure 5. Obviously properties (i) and (iii) are satisfied. To see (ii), note that any agent contained in the first or last block of δ-tubes cannot leave its tube (and thus the L-δ-chain) by 3δ or more. On the other hand, none of the inner blocks' agents is able to even cross the preceding or the following block of six tubes, since their total length is larger than its budget.

Arc Replacement of ζ-Segments. Next we decouple any pair of clause agents (stationed at nodes $c_{i,j}$, $c_{j,k}$) that can directly go to the same literal node $l_{j,c}$ (so as not to allow them to transport the package via clause node with their

[5] Non-returning clause agents can do this if they are 3-resource-augmented; and we can prevent it for $(3-\varepsilon)$-resource-augmentation by setting ζ such that $(3-\varepsilon)\cdot(1+\zeta) \ll 3$ (in fact the value of ζ will be the same as for *Returning* BudgetedDelivery, but we will use different bounds in the proof).

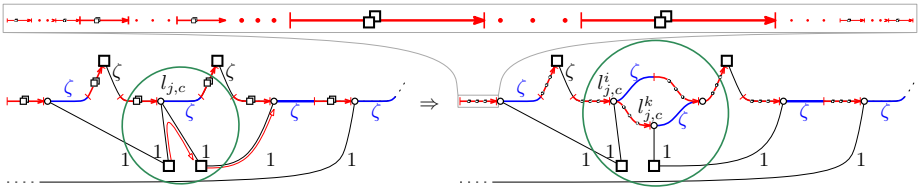

Figure 6: (top-to-bottom) We replace δ-tubes in $G(F)$ by L-δ-chains in $G'(F)$.
(left-to-right) We replace each ζ-segment connected to two clause agents by two parallel arcs.

augmented budgets, depicted in red in Figure 6 (left)). We replace the adjacent ζ-segment by two small arcs which represent alternative ways over which the package can be transported. Each arc consists of one L-δ-chain and of one ζ-segment, see Figure 6.
The *inner arc* begins with the ζ-segment – whose beginning $l^i_{j,c}$ can be reached through an edge of length 1 by the first clause agent (stationed at $c_{i,j}$) – and ends with the L-δ-chain. The *outer arc* first has the L-δ-chain and then the ζ-segment. The node in between these two parts, denoted by $l^k_{j,c}$, is connected via an edge of length 1 to the second clause agent's starting position $c_{j,k}$.

We conclude the replacement with three remarks: Firstly, it is easy to see that the described operation respects the planarity of the graph. Secondly, we are able to give values for L and δ in the next paragraph such that a single clause agent is still both necessary and (together with agents inside the newly created adjacent L-δ-chain) sufficient to transport a package over one of the parallel arcs from left to right. Finally, the clause agent starting at $c_{i,j}$ is no longer able to meet the clause agent starting at $c_{j,k}$.

Budgets and Edge Weights. Recall that our agents have the following budgets: *separating agents* have a budget according to their position in the L-δ-chain, *variable agents* a budget of 2ζ and *clause agents* a budget of $2(1+\zeta)$.[6] Now these budgets are γ-resource-augmented, with $\gamma < 3$. We would like to prevent clause and variable agents from crossing L-δ-chains or even meeting inside of them, hence we set $L := 9$, which shall exceed the augmented budget of every agent by a factor of more than 2. Furthermore we don't want separating agents to help out too much outside of their residential chain, hence we set $\delta := \frac{\zeta}{9}$. A resource-augmented separating agent can thus walk only as far as $3\delta = \frac{\zeta}{3}$ to the outside of the chain. In particular, separating agents cannot transport the package over a ζ-segment. At the same time, if the agents were able to deliver the package in $G(F)$, then they can do so in $G'(F)$:

Proposition 4. *If a planar 3CNF F is satisfiable, then in the corresponding delivery graph $G'(F)$, the agents can collectively deliver the package from s to t*

[6] In the *Non-Returning* version, variable agents have a budget of ζ and clause agents a budget of $1+\zeta$.

and return to their respective starting positions.

Proof. In Proposition 1 we have seen how the package can be transported in the original delivery graph $G(F)$. In the modified delivery graph $G'(F)$, *variable agents* and *clause agents* do exactly the same as their counterparts in $G(F)$, and *separating agents* help wherever needed. □

Next we choose ζ such that an augmented clause agent stationed at a clause node $c_{i,j}$ is not able to transport the package from $l_{i,c}$ to $l_{j,c}$, not even in collaboration with the separating agents that can reach the two literal nodes. We set $\zeta := \frac{\varepsilon}{6-\varepsilon}$. The edges $\{c_{i,j}, l_{i,c}\}$, $\{c_{i,j}, l_{j,c}\}$ have length 1. In each edge, separating agents can help by at most $3\delta = \frac{\zeta}{3}$, leaving at least a distance of $1 - \frac{\zeta}{3}$ for the clause agent to cover. First note that for $0 < \varepsilon < 1$, we have $\zeta = \frac{\varepsilon}{6-\varepsilon} < \frac{\varepsilon}{5} < \frac{2\varepsilon}{3}$ and $(6-\varepsilon) > 3(2-\varepsilon)$. Hence a γ-resource-augmented clause agent has a budget of only $\gamma \cdot 2(1+\zeta) = 2(2-\varepsilon)(1+\zeta) = 2(2-\varepsilon+\frac{(2-\varepsilon)\varepsilon}{6-\varepsilon}) < 2(2-\frac{2\varepsilon}{3}) < 2(2-\zeta) < 4 \cdot (1-\frac{\zeta}{3})$, and thus cannot transport the package via its clause node and return home in the end.[7] We get:

Proposition 5. *It is not possible for $(2-\varepsilon)$-resource-augmented agents to deliver the package from s to t via any clause node of the delivery graph $G'(F)$.*[8]

Finally, we need to show that – as in Proposition 3 – a transport over a ζ-segment needs the corresponding variable or clause agent. Additionally, any feasible schedule with augmented budgets should correspond to a feasible schedule with original budgets.

Proposition 6. *Assume that there is a schedule in which γ-resource-augmented agents ($\gamma < 3$) collectively deliver the package from s to t in the delivery graph $G'(F)$. Then in each ζ-segment that the package is delivered over, the schedule uses the corresponding variable agent or the corresponding clause agent. Furthermore the schedule can be transformed into a feasible schedule with the original budgets.*

Proof. By Proposition 5 we know that the package cannot be transported over any of the clause nodes. Recall that γ-resource-augmented separating agents inside δ-tubes can neither single-handedly nor collectively (starting from both sides) transport the package over a ζ-segment, since they can move outside of their residential L-δ-chain by at most $3\delta = \frac{\zeta}{3}$. Furthermore, clause agents and variable agents are not able to meet each other anywhere in the graph, since they are pairwise separated by at least one L-δ-chain and do not have enough energy (even with the resource-augmented budgets) to reach the middle point of one of these chains.

[7] For non-returning agents we use (for $\varepsilon < 2$) the inequalities: $\zeta = \frac{\varepsilon}{6-\varepsilon} < \frac{\varepsilon}{4} < \frac{\varepsilon}{2}$ and $(6-\varepsilon) > 2(3-\varepsilon)$. Hence a non-returning γ-resource-augmented clause agent has a budget of $\gamma(1+\zeta) = (3-\varepsilon)(1+\zeta) = 3-\varepsilon+\frac{(3-\varepsilon)\varepsilon}{6-\varepsilon} < 3-\frac{\varepsilon}{2} < 3-\zeta = 3\cdot(1-\frac{\zeta}{3})$, and thus cannot transport the package via its clause node.

[8] For the *Non-Returning* setting the proposition holds for $(3-\varepsilon)$-resource-augmented agents.

In conclusion, we know that the package needs to be transported along the true- and false-paths and without loss of generality we assume that this happens in a strictly monotone movement. Now in each ζ-segment the package is transported across in the schedule with $(2-\varepsilon)$-resource-augmented budget, a variable agent or a clause agent is necessary. Since these agents cannot meet each other, such an agent must pick up the package from a separating agent of the preceding L-δ-chain and hand it over to a separating agent of the following L-δ-chain. Even with a non-augmented budget, said agent would be able to pick up and hand over the package at the end and the start of these chains. Additionally, separating agents are able to transport the package from left to right across their L-δ-chain without a resource augmentation of their budgets. Together this yields a solution for delivery on $G'(F)$ without any resource-augmented budgets. □

Lemma 2 (Resource-augmented Returning BudgetedDelivery)**.** *A planar 3CNF F is satisfiable if and only if it is possible to deliver a package with $(2-\varepsilon)$-resource-augmented agents from s to t in the corresponding delivery graph $G'(F)$, such that the agents are still able to reach their starting point in the end.*

Proof. "⇒" This direction has been shown in Proposition 4.

"⇐" Assume that the package can be delivered from s to t. From Proposition 5 it follows that the package has to be transported through the true- and false-paths. By Proposition 6 we know that the schedule thus can be transformed into a schedule of agents with non-augmented budgets, where in each ζ-segment that the package is delivered over, the corresponding variable agent or the corresponding clause agent is used.

We show that there is a bijective mapping into a feasible schedule in the original graph $G(F)$, which by Lemma 1 gives us a satisfiable assignment for F. Consider the movement of the individual agents: First of all, we let every *variable agent* in $G(F)$ do the same work as its counterpart in $G'(F)$ and vice versa. Now consider the *separating agents* of any δ-tube in $G(F)$ which corresponds to a L-δ-chain in $G'(G)$. We let these agents collectively transport the package from left to right over their δ-tube in $G(F)$ if and only if the agents in the corresponding chain in $G'(F)$ transport the package over their L-δ-chain. Finally, we let corresponding *clause agents* in both graphs go to the same ζ-segment (both to their first segment, both to their second segment, or both to neither). Hence agents in $G(F)$ can just "copy" the movements of their respective counterparts in $G'(F)$. □

It is easy to see that the same arguments work for *Non-Returning* Budgeted-Delivery as well: In the proof, we simply replace the use of Lemma 1 by referring to Corollary 1 and replace the estimations in the proof of Proposition 5 with the corresponding estimations for non-returning $(3-\varepsilon)$-resource-augmented agents, given in Footnote 7.

Corollary 2 (Resource-augmented Non-Returning BudgetedDelivery)**.** *A planar 3CNF F is satisfiable if and only if it is possible to deliver a package with*

$(3-\varepsilon)$-*resource-augmented agents from* s *to* t *in the corresponding delivery graph* $G'(F)$.

Compare the new delivery graph $G'(F)$ with the original graph $G(F)$. The only topological changes we introduced with our replacements were the parallel arcs replacing the ζ-segments reachable by two clause nodes. We have already seen that this change respected the planarity of the delivery graph. Relevant changes to the edge weights and agent numbers, on the other hand, were added by replacing δ-tubes with L-δ-chains: Each chain consists of blocks of six δ-tubes of exponentially increasing size, hence we need a logarithmic number of tubes per chain, namely $\mathcal{O}\left(\log \frac{L}{\delta}\right)$ many. We have fixed the values of L and δ to $L = 9$ and $\delta = \frac{\zeta}{9}$. With $\zeta^{-1} = \frac{9}{\varepsilon} - 1 \in \Theta(\varepsilon^{-1})$ we get $\mathcal{O}\left(\log \frac{L}{\delta}\right) = \mathcal{O}(\log(\zeta^{-1})) = \mathcal{O}(\log(\varepsilon^{-1}))$ many agents per chain. The number of chains is clearly polynomially bounded by the number of variables and clauses and the edge weights depend on ε only as well. Hence we conclude:

Theorem 6 (Inexistence of a better resource augmentation for BudgetedDelivery)**.** *There is no polynomial-time* $(2-\varepsilon)$*-resource-augmented algorithm for* Returning BudgetedDelivery *and no* $(3-\varepsilon)$*-resource-augmented algorithm for* Non-Returning BudgetedDelivery*, unless* P = NP.

6. Discussion

We gave a polynomial time algorithm for the returning variant of the problem on trees, as well as a best-possible resource-augmented algorithm for general graphs. On the other hand, we have shown that BudgetedDelivery (both the returning and non-returning version) is NP-hard, even on planar graphs, or even if we allow "small" resource augmentation. Our hardness bounds on the required resource augmentation are tight and complement the previously known algorithm [3] for the non-returning case.

Our results show that the returning version of BudgetedDelivery becomes hard when transitioning from trees to planar graphs. Naturally, it is interesting to investigate, from the graph topology point of view, when exactly the transition from easy to hard happens. At the same time, it is not only the topology that makes the problem hard. In the following we have a light look at various aspects that make the problem easy or hard.

Intermediate graph classes. We have seen that the *Returning* BudgetedDelivery is solvable in polynomial time on trees. Here, we extend this positive result to graphs, which correspond to the union of internally vertex-disjoint s-t-paths. Observe that a special case of such a graph is a ring (also called a cycle), which is one of the simplest connected planar graphs that is not a tree.

Theorem 7. *There is an* $\mathcal{O}(n + k \log k)$*-time algorithm for* Returning BudgetedDelivery *on graphs consisting of internally vertex-disjoint* s-t*-paths.*

Proof. Denote by $P_1, \ldots, P_l$ the internally vertex-disjoint s-t-paths of the input graph G. Observe that if there is a solution to the given instance, then the packet travels from s to t along exactly one of the l paths. Let us call such a path a *delivery* path. Hence, we can try each of the l paths, and check whether the agents can deliver the packet along the path. This problem is quite similar to BudgetedDelivery on the line, with the exception that now the agents that are not on the path (along which the delivery happens) can enter the path from two vertices – from s and from t. Observe that when searching for a solution, we can assume that there is always at most one agent that is not on the delivery path and that enters the path via s. If there were two or more such agents, then we could simply keep, among all such agents, only the agent that reaches the farthest vertex from s along the path. By similar arguments, we can assume that at most one agent enters the delivery path via t. We can thus enumerate all possible agents that can help to deliver the package along the delivery path. In detail, this can be done as follows.

As in the case of trees, we can find the distances from all agents' starting positions to s and t in time $\mathcal{O}(n)$ using breadth-first search from both s and t. Hence, for each agent we know whether it can reach s and whether it can reach t, and we also know how much budget the agent has left, once it reaches, respectively, s and t. For any path P_i, we denote by k_i the number of agents with starting position p_j on P_i and keep track of the two agents $a_{s,i,1}$, $a_{s,i,2}$ with the two highest remaining budgets after moving to s, i.e. we take $a_{s,i,1} := \arg\max_j \{B_j - d(s, p_j) \mid p_j \in P_i \text{ and } d(s, p_j) \geq B_j\}$ and similarly $a_{s,i,2} := \arg\max_j \{B_j - d(s, p_j) \mid p_j \in P_i \text{ and } d(s, p_j) \geq B_j \text{ and } j \neq a_{s,i,1}\}$. In the same way, we store the two agents $a_{t,i,1},\ a_{t,i,2}$ with the highest remaining budget after moving to t. Note that the two latter are not necessarily disjoint from the former, and these agents may not exist at all (if this is the case, we simply discard them from the following discussion). Finally, among all agents $a_{s,i,1},\ a_{s,i,2},\ i = 1, \ldots, l$ we keep the set of the four agents with the highest remaining budget after they reach s, which we denote by A_s. We do the same at t to get A_t, the set of agents with the highest remaining budget at t.

Assume that we are interested in whether the package can be delivered from s to t along the path P_i: Applying the result for trees and lines (Theorem 1), it is clear how to solve the decision problem for P_i – by only using agents with starting position on P_i – in time $\mathcal{O}(k_i \log k_i)$. However, the package might get advanced from s towards t on P_i by some *external* agent j originating from another path (i.e. an agent with $p_j \notin P_i$); the package might also be picked up by such an external agent and taken to t. Without loss of generality, this will be done – if at all – by the external agent $a_{s,j,1},\ j \neq i$ which has the highest remaining budget at s, or *if this agent is also the agent which has the highest remaining budget at* t, by the external agent $a_{s,j,2},\ j \neq i$ with the second highest remaining budget (and the same observation holds for t).

Clearly, these candidates are from the sets A_s and A_t, and testing all pairwise combinations yields a running time of $\mathcal{O}(|A_s||A_t|k_i \log k_i) = \mathcal{O}(k_i \log k_i)$. Hence we can simply go over all paths P_i in time $\mathcal{O}\left(\sum_{i=1}^{l} k_i \log k_i\right) \subseteq \mathcal{O}(k \log k)$, and check whether the package can be delivered along P_i. □

This positive result raises the question of whether BUDGETEDDELIVERY is polynomial-time solvable for other subclasses of planar graphs such as series-parallel graphs or outerplanar graphs. So far, this remains an intriguing open problem.

Fixed agent order. Chalopin et.al. [3] gave a polynomial algorithm for the *Non-Returning* version under the assumptions that (i) the order in which the agents move is fixed and (ii) the package can only be handed over at vertices. Using a dynamic program, we are able to drop assumption (ii), allowing handovers within edges. Our result holds for both versions of BUDGETEDDELIVERY.

Theorem 8. BUDGETEDDELIVERY *is solvable in time* $\mathcal{O}(k(n+m)(n \log n+m))$ *if the agents are restricted to a fixed order in which they move.*

Proof. If there is a feasible schedule, we can compute it in a breadth-first search-like fashion where we proceed agent by agent and update reachable regions of the graph on-the-fly: Each agent can either not help in the schedule or it can transport the package from a pickup location to a drop-off location. We show that we can restrict drop-offs to meaningful places such that for each agent the set of all possible pickup locations is bounded by $n+m$. This limitation to only one of the potentially infinitely many handover points inside each edge allows us to use dynamic programming and to proceed by induction:

Denote the agents in the schedule order by $a_1, \ldots, a_\ell$. The first agent a_1 can pick up the package at s only, hence there is only one possible pick-up location. If a_1 wants to drop off the package at a vertex, there are at most n choices of where to do so. We mark all the vertices which a_1 can reach from s while still being able to return home. Now assume a_1 wants to drop off the package inside an edge $e = \{u, v\}$. This means that e can be reached by a_1, hence without loss of generality the vertex u is marked. If v is marked as well, then a_1 should *not* drop the package inside e, since the package has to be picked up later, which could just as well be done at either u or v. Otherwise a_1 should bring the package *as far as possible into the edge* (since if a later agent a_i wants to pick up the package, it can pick it up at u or come in via v). We mark this point inside the edge and store its distance from u. We now restrict ourselves to these at most $n + m$ described drop-off locations.

An agent $a_i,\ i > 1$ can pick up the package at s or at any previous drop-off location. By induction there are at most $n + m$ many such locations. Now we first check whether a_i can pick up the package somewhere and deliver it to any not yet marked vertex. If so, we mark this vertex (and the number of marked vertices stays at most n). Next we check whether a_i can bring the package into an edge $e = \{u, v\}$ for which (without loss of generality) u is marked and v is not. Check whether the point inside the edge which is furthest from u – and still can be reached by a_i – has larger distance to u than a previously marked point. If so, delete the old point (if any) and mark the new point and store its distance from u. The number of marked edges stays at most m.

If at some point we mark the vertex t, we are done. Since each agent i has at most $n + m$ pick-up locations to consider, we can compute all new marks by

computing the ellipsoid $\mathcal{E}(p_i, l, B_i)$ for every old mark l, which we can do by running Dijkstra's shortest path algorithm once from p_i and once from each old mark. Hence we require time $\mathcal{O}((n+m) \cdot (n \log n + m))$ per agent. □

Corollary 3. *For a constant number of agents k,* BudgetedDelivery *is solvable in time $\mathcal{O}(poly(n, m))$ by brute forcing the order of the agents.*

Future work. An interesting open problem is to understand collaborative delivery of multiple packages at once. For example, the complexity of the problem on paths remains open. In this setting, it may be reasonable to constrain the number of agents, the number of packages, or the ability of transporting multiple packages at once, in order to allow for efficient algorithms. Also, in general graphs, the problem may not become easy if the order in which agents move is fixed.

References

[1] P. Flocchini, G. Prencipe, N. Santoro, Distributed Computing by Oblivious Mobile Robots, Morgan & Claypool, 2012.

[2] J. Anaya, J. Chalopin, J. Czyzowicz, A. Labourel, A. Pelc, Y. Vaxès, Convergecast and broadcast by power-aware mobile agents, Algorithmica 74 (1) (2016) 117–155.

[3] J. Chalopin, S. Das, M. Mihalák, P. Penna, P. Widmayer, Data delivery by energy-constrained mobile agents, in: 9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics ALGOSENSORS'13, 2013, pp. 111–122.

[4] J. Chalopin, R. Jacob, M. Mihalák, P. Widmayer, Data delivery by energy-constrained mobile agents on a line, in: 41st International Colloquium on Automata, Languages, and Programming ICALP'14, 2014, pp. 423–434.

[5] D. L. Applegate, R. E. Bixby, V. Chvatal, W. J. Cook, The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics), Princeton University Press, Princeton, NJ, USA, 2007.

[6] J. Edmonds, E. L. Johnson, Matching, euler tours and the chinese postman, Mathematical Programming 5 (1) (1973) 88–124.

[7] P. Panaite, A. Pelc, Exploring unknown undirected graphs, Journal of Algorithms 33 (2) (1999) 281–295.

[8] S. Albers, M. R. Henzinger, Exploring unknown environments, SIAM Journal on Computing 29 (4) (2000) 1164–1188.

[9] M. A. Bender, D. K. Slonim, The power of team exploration: Two robots can learn unlabeled directed graphs, in: 35th Symposium on Foundations of Computer Science, FOCS'94, 1994, pp. 75–85.

[10] M. Betke, R. L. Rivest, M. Singh, Piecemeal learning of an unknown environment, Machine Learning 18 (2) (1995) 231–254.

[11] B. Awerbuch, M. Betke, R. L. Rivest, M. Singh, Piecemeal graph exploration by a mobile robot, Information and Computation 152 (2) (1999) 155–172.

[12] C. A. Duncan, S. G. Kobourov, V. S. A. Kumar, Optimal constrained graph exploration, in: 12th ACM Symposium on Discrete Algorithms, SODA'01, 2001, pp. 807–814.

[13] P. Fraigniaud, L. Gąsieniec, D. R. Kowalski, A. Pelc, Collective tree exploration, Networks 48 (3) (2006) 166–177.

[14] M. Dynia, M. Korzeniowski, C. Schindelhauer, Power-aware collective tree exploration, in: 19th International Conference on Architecture of Computing Systems, ARCS'06, 2006, pp. 341–351.

[15] M. Dynia, J. Łopuszański, C. Schindelhauer, Why robots need maps, in: 14th International Colloquium on Structural Information and Communication Complexity, SIROCCO'07, 2007, pp. 41–50.

[16] S. Das, D. Dereniowski, C. Karousatou, Collaborative exploration by energy-constrained mobile robots, in: 22th International Colloquium on Structural Information and Communication Complexity SIROCCO'15, 2015, pp. 357–369.

[17] D. Dereniowski, Y. Disser, A. Kosowski, D. Pająk, P. Uznański, Fast collaborative graph exploration, Information and Computation 243 (2015) 37–49.

[18] C. Ortolf, C. Schindelhauer, Online multi-robot exploration of grid graphs with rectangular obstacles, in: 24th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA'12, 2012, pp. 27–36.

[19] E. D. Demaine, M. Hajiaghayi, H. Mahini, A. S. Sayedi-Roshkhar, S. Oveisgharan, M. Zadimoghaddam, Minimizing movement, ACM Trans. Algorithms 5 (3) (2009) 1–30.

[20] D. Bilò, Y. Disser, L. Gualà, M. Mihalák, G. Proietti, P. Widmayer, Polygon-constrained motion planning problems, in: 9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics ALGOSENSORS'13, 2013, pp. 67–82.

[21] A. Bärtschi, J. Chalopin, S. Das, Y. Disser, D. Graf, J. Hackfeld, A. Labourel, P. Penna, Energy-efficient Delivery by Heterogeneous Mobile Agents, in: 34th International Symposium on Theoretical Aspects of Computer Science STACS'17, 2017, to appear.

[22] A. Bärtschi, J. Chalopin, S. Das, Y. Disser, D. Graf, J. Hackfeld, A. Labourel, P. Penna, Energy-efficient Delivery by Heterogeneous Mobile Agents, CoRR arXiv:1610.023610.

[23] J. Czyzowicz, K. Diks, J. Moussi, W. Rytter, Communication problems for mobile agents exchanging energy, in: 23rd International Colloquium on Structural Information and Communication Complexity SIROCCO'16, 2016.

[24] D. Lichtenstein, Planar formulae and their uses, SIAM Journal on Computing 11 (2) (1982) 329–343.