Collective Fast Delivery by Energy-Efficient Agents

Andreas Bärtschi
Department of Computer Science, ETH Zürich, Switzerland
andreas.baertschi@inf.ethz.ch

Daniel Graf
Department of Computer Science, ETH Zürich, Switzerland
daniel.graf@inf.ethz.ch

Matúš Mihalák
Department of Data Science and Knowledge Engineering, Maastricht University, Netherlands
matus.mihalak@maastrichtuniversity.nl

Abstract

We consider k mobile agents initially located at distinct nodes of an undirected graph (on n nodes, with edge lengths). The agents have to deliver a single item from a given source node s to a given target node t. The agents can move along the edges of the graph, starting at time 0, with respect to the following: Each agent i has a *weight* ω_i that defines the rate of energy consumption while travelling a distance in the graph, and a *velocity* υ_i with which it can move.

We are interested in schedules (operating the k agents) that result in a small *delivery time* $\mathcal{T}$ (time when the item arrives at t), and small *total energy consumption* $\mathcal{E}$. Concretely, we ask for a schedule that: either (i) Minimizes $\mathcal{T}$, (ii) Minimizes lexicographically $(\mathcal{T}, \mathcal{E})$ (prioritizing fast delivery), or (iii) Minimizes $\epsilon \cdot \mathcal{T} + (1-\epsilon) \cdot \mathcal{E}$, for a given $\epsilon \in (0, 1)$.

We show that (i) is solvable in polynomial time, and show that (ii) is polynomial-time solvable for uniform velocities and solvable in time $\mathcal{O}(n + k \log k)$ for arbitrary velocities on paths, but in general is NP-hard even on planar graphs. As a corollary of our hardness result, (iii) is NP-hard, too. We show that there is a 2-approximation algorithm for (iii) using a single agent.

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

Keywords and phrases delivery, mobile agents, time/energy optimization, complexity, algorithms

Digital Object Identifier 10.4230/LIPIcs.MFCS.2018.56

Funding This work was partially supported by the SNF (project 200021L_156620, Algorithm Design for Microrobots with Energy Constraints).

1 Introduction

Technological development has allowed for low-cost mass production of small and simple mobile robots. Autonomous vacuum cleaners, mowers, or drones are some of the best known examples. There are attempts to deploy such autonomous *agents* to deliver physical goods – *packages* [24, 26]. In the future, for delivering over longer distances, a *swarm* of such autonomous agents is a likely option to be adapted, since the energy supply of the agents is limited, or the agents are simply required to operate locally, or simply because the usage of some agents is more costly than others. A careful cooperation and planning of the agents is thus necessary to provide energy, time, and cost efficient delivery. This leads to plentiful optimization problems regarding the operation of the agents.

43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018).
Editors: Igor Potapov, Paul Spirakis, and James Worrell; Article No. 56; pp. 56:1–56:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Here we consider the problem of delivering a single package as quickly as possible from a source node s to a target node t in a graph $G = (V, E)$ with edge lengths by a team of k agents. The agents have individual velocities, with which they can move along the edges of the graph, and also an energy-consumption rate for a travelled unit distance. The goal is to design *centralized algorithms* to coordinate the agents such that the package is delivered from s to t in an efficient way. In the literature, delivery problems focusing solely on energy efficiency have been studied. One research direction considers every agent to have an initial amount of energy (battery) that restricts the agents' movements [1, 11]. The decision problem of whether the agents can deliver the package has been shown to be strongly NP-hard on planar graphs [6, 7] and weakly NP-hard on paths [12], and it remains NP-hard on general graphs even if the agents can exchange energy [13]. The second research direction considers every agent to have unlimited energy supply, and an individual energy-consumption rate per travelled distance [8, 9]. The problem of delivering the package and minimizing the total energy consumption can be solved in time $\mathcal{O}(k + n^3)$ [8].

In this paper, we primarily focus on delivering the package in a quickest possible way, and only secondarily on the total energy that is consumed by the agents. This has not been, to the best of our knowledge, studied before. Specifically, we consider the algorithmic problem of finding a delivery schedule that: (i) minimizes the delivery time, (ii) minimizes the delivery time using the least amount of energy, and (iii) minimizes a linear combination of delivery time and energy consumption.

Our model. We are given an undirected graph $G = (V, E)$ on $n = |V|$ nodes. Each edge $e \in E$ has a positive *length* l_e. The length of a path is the sum of the lengths of its edges. We consider every edge $e = \{u, v\}$ to consist of infinitely many points, where every point is uniquely characterized by its distance from u, which is between 0 and l_e. We consider every such point to subdivide the edge $\{u, v\}$ into two edges of lengths proportional to the position of the point on the edge. The distance $d_G(p, q)$ between two points p and q (nodes or points inside edges) of the graph is the length of a shortest path from p to q in G. There are k mobile agents initially placed on nodes $p_1, \ldots, p_k$ of G. Every agent $i = 1, \ldots, k$ has a *weight* $0 \leq \omega_i < \infty$ and a *velocity* $0 < \upsilon_i \leq \infty$. Agents can traverse the edges of the graph. To traverse an edge e (in either direction), agent i needs time l_e/υ_i and $\omega_i \cdot l_e$ units of energy.

Furthermore there is a single package, initially (at time 0) placed on a source node s, which has to be delivered to a given target node t. Each agent can walk from its current location to the current location of the package (along a path in the graph), pick the package up, carry it to another location (a point of the graph), and drop it there. From this moment, another agent can pick up the package again. Only the moving in the graph takes time – picking up the package and dropping it off is done instantaneously. (The time spent by the package being dropped at a point until picked up again is, however, taken into account.)

We call a schedule that operates the agents such that the package is delivered a *solution*. In such a schedule S, we denote by $d_i(S)$ the total distance travelled by agent i, and by $d_i^*(S)$ the distance travelled by agent i while carrying the package. The total energy consumption of the solution is thus $\mathcal{E}(S) = \sum_{i=1}^k \omega_i \cdot d_i(S)$ and the time needed to deliver the package is given by $\mathcal{T}(S) = \sum_{i=1}^k d_i^*(S)/\upsilon_i +$ (the overall time the package is not carried). *Fast and energy-efficient* Delivery is the optimization problem of finding a solution that has small delivery time $\mathcal{T}$ as well as total energy consumption $\mathcal{E}$. In particular, we study the following three objectives (see Figure 1 for illustration):

(i) Minimize the delivery time $\mathcal{T}$.

(ii) Lexicographically minimize the tuple $(\mathcal{T}, \mathcal{E})$, i.e. among all solutions with minimum $\mathcal{T}$, find a solution that has minimum energy consumption $\mathcal{E}$.

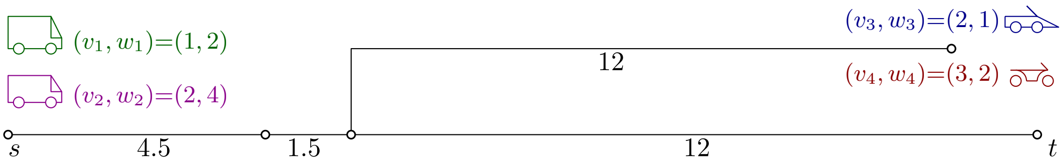

Figure 1 Example for optima of variants of fast *and* energy-efficient DELIVERY:
(ii) Using agents 2 and 4, we get $(\mathcal{T}, \mathcal{E}) = (\max\{6/2, 12/3\} + 12/3,\ 4 \cdot 6 + 2 \cdot 12 + 2 \cdot 12) = (8, 72)$.
(iii) For $\epsilon = \frac{4}{5}$, using agents 1 and 4, we get $\frac{4}{5}\mathcal{T} = \frac{4}{5}\,(\max\{4.5/1, (12+1.5)/3\} + (1.5+12)/3)$ and $\frac{1}{5}\mathcal{E} = \frac{1}{5}(4.5 \cdot 2 + (12+1.5) \cdot 2 + (1.5+12) \cdot 2)$ for a combined total of $\frac{4}{5}\,(4.5+4.5) + \frac{1}{5}(9+27+27) = 19.8$.
(iv) Using agents 1 and 3, we get $(\mathcal{E}, \mathcal{T}) = (2 \cdot 6 + 1 \cdot 12 + 1 \cdot 12,\ \max\{6/1, 12/2\} + 12/2) = (36, 12)$.

(iii) Minimize a convex combination $\epsilon \cdot \mathcal{T} + (1-\epsilon) \cdot \mathcal{E}$, for some given value $\epsilon \in (0, 1)$.
Recent parallel work studied the following complementary – energy focused – variants:
(iv) Lexicographically minimize the tuple $(\mathcal{E}, \mathcal{T})$, i.e. prioritize the minimization of $\mathcal{E}$ [10].
(v) Minimize the energy consumption $\mathcal{E}$ [8, 9].
In all variants it is natural to (without loss of generality) only consider simple paths as the trajectory of the package, i.e., if at times t_1, t_2 $(0 \le t_1 \le t_2 \le \mathcal{T})$ the package is at the same position p, then it remains at position p for the time in-between $(\forall t \in [t_1, t_2])$. We will make this assumption throughout this paper.

Our contribution. First, in Section 2, we prove for the first time that optimum solutions exist for all mentioned variants of DELIVERY (while previous work on (iv) and (v) implicitly assumed this). Then, in Section 3, we investigate the problem of minimizing the delivery time $\mathcal{T}$ only. We call this optimization problem FASTDELIVERY and show that there is a polynomial-time dynamic program of time complexity $\mathcal{O}(k^2|E| + k|V|^2 + \text{APSP}) \subseteq \mathcal{O}(k^2n^2 + n^3)$, where $\mathcal{O}(\text{APSP})$ is the running time of an all-pair shortest path algorithm for undirected graphs.

In Section 4, we study FASTEFFICIENTDELIVERY, prioritizing the delivery time $\mathcal{T}$ over the energy consumption $\mathcal{E}$. We first show that the problem can be solved in polynomial time for uniform velocities. However, we prove the problem to be NP-hard for general velocities even on planar graphs. We therefore consider the restricted graph class of *paths*, in which we can decompose the problem into uniform velocity instances. For each such instance, we establish a characterization of handover points. Using geometric point-line duality [18] and dynamic planar convex hull techniques [4], we give an $\mathcal{O}(n + k \log k)$ algorithm for paths.

In Section 5, we show that for arbitrary given weights $\epsilon \in (0, 1)$, the minimum convex combination $\epsilon \cdot \mathcal{T} + (1-\epsilon) \cdot \mathcal{E}$ can be 2-approximated by a single agent, while NP-hardness follows from an adaptation of the hardness proof in the preceding section. We call the task of minimizing the convex combination COMBINEDDELIVERY. Finally, in Section 6 we discuss several extended models to which our approach can be generalized. Due to the limited space, some proofs are omitted, but are provided in a thesis on several variants of DELIVERY [5].

Comparison to related work. Among the earliest problems related to DELIVERY are the Chinese Postman Problem [19] and the Traveling Salesman Problem [2], in which a single agent has to visit multiple destinations located in edges or nodes of the graph, respectively. The latter has given rise to a class of problems known as Vehicle Routing Problems [25], which are concerned with the distribution of goods by a fleet of (homogeneous) vehicles under additional hard constraints such as time windows. Minimizing the total or the maximum travel distance of a group of agents for several tasks such as the formation of configurations [17] or the visit of designated arcs [20] have been studied for identical agents as well. Energy-efficient Delivery (without optimization of delivery time) has been recently introduced [8] for an arbitrary number of packages, with handovers restricted to take place at nodes of the graph only. This setting turns out to be NP-hard, but can be solved in polynomial-time for a single package, in which case the restriction of handovers to nodes becomes irrelevant (there is always an optimal solution which does not use any in-edge handovers). To the best of our knowledge, this present paper and a parallel work [10] on variant (iv) are the only ones studying the Delivery problem with agents which have different velocities. Similar to our approach, the latter studies a uniform weight setting first. The uniform weight result is then used as a subroutine in a dynamic program for general weights. Our hardness result shows that such an approach (combination of uniform velocities) is not possible for FastEfficientDelivery, even on planar graphs. Finally, mobile agents with distinct maximal velocities have been getting attention in areas such as searching [3], walking [14] and patrolling [15].

2 Preliminaries

We first formally establish that optimum solutions for all variants of efficient Delivery exist. To this end, each solution which operates agents $i_1, i_2, \ldots, i_\ell$ in this order can be represented by the drop-off locations of these agents only (note that for two consecutive agents i, j, the drop-off location of agent i, denoted by q_i^-, corresponds to the pick-up location of agent j, denoted by q_j^+). Since we allow in-edge handovers, there are infinitely many solutions – however, these can be divided into *finitely* many topologically compact sets. As $\mathcal{E}, \mathcal{T}$ act as continuous functions on these sets, we have in each set a minimum solution.

▶ **Theorem 1** (Existence of optimum solutions). *There exists an optimum solution minimizing the delivery time $\mathcal{T}$ (the energy consumption $\mathcal{E}$, or $\epsilon \cdot \mathcal{T} + (1-\epsilon) \cdot \mathcal{E}$, $(\mathcal{T}, \mathcal{E})$, $(\mathcal{E}, \mathcal{T})$, respectively).*

3 Optimizing delivery time only

Throughout this section, we assume that all agents have weight $\omega_i = 0$. Hence in all three variants of fast energy-efficient Delivery, $\mathcal{E} = 0$ and we are after a solution for delivery with earliest-possible delivery time. We show that FastDelivery is polynomial-time solvable, due to the following characterization of optimum solutions (which exist by Theorem 1):

▶ **Lemma 2.** *For every instance of* FastDelivery*, there is an optimum solution in which (i) the velocities of the involved agents are strictly increasing, (ii) no involved agent arrives at its pick-up location earlier than the package (carried by the preceding agent), and (iii) if more than one agent is involved in transporting the package over an edge $\{u, v\}$ in direction from u to v, then only the first involved agent will ever visit u.*

Proof. All three properties can be shown by exchange arguments. Taking any optimum solution, we turn it into an optimum solution that adheres to the three properties as follows:

(i) Label the agents $1, 2, \ldots, i, \ldots$ in the order in which they transport the package. Let i be the first agent such that $\upsilon_i \geq \upsilon_{i+1}$. Now we can simply replace agent $i+1$ by letting agent i travel on the same trajectory on which $i+1$ transported the package; and by doing so, we don't increase the delivery time.

(ii) Let i be the first agent that has to wait at its pick-up location for the package to arrive. Instead of waiting, we let i proceed on the original trajectory of the package towards s until it meets the preceding agent $i-1$. Handing over the package at this new spot cannot

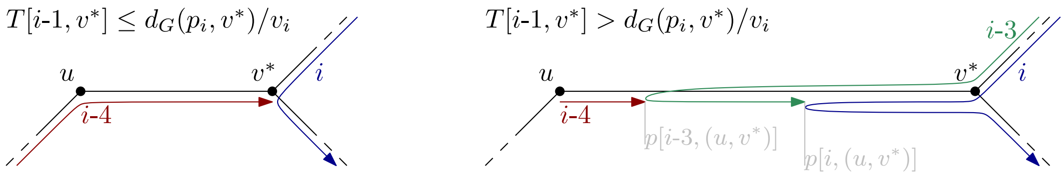

Figure 2 Examples for cases a) and b): (left) Agent i picks up the package at node v^*. (right) Agent i picks up the package inside the edge (u, v^*) at the earliest possible time.

increase the delivery time $\mathcal{T}$, as $v_{i-1} < v_i$ (we only increase velocities along the trajectory). However, $\mathcal{T}$ might remain constant if this increase in velocity is countered by a longer waiting time of the package at the handover to agent $i + 1$.

(iii) Assume that multiple agents bring the package from u to v over the edge $\{u, v\}$, by visiting u first. By assumption (i) the last such agent i has the highest velocity and thus agent i can just as well pick up the package at u without the help of the other agents. ◀

▶ **Corollary 3.** *After a preprocessing step of time $\mathcal{O}(k + |V|)$ – in which we remove in each node all but the agent with maximum velocity v_i – we may assume that $k \leq |V|$.*

Towards a dynamic program. Making use of characterization (i) of Lemma 2, we relabel the agents such that $v_1 \leq v_2 \leq \ldots \leq v_k$. We can then look at subproblems where we only use the first $i - 1$ among all k agents. Assume node v^* is the first node that the new agent i (starting at p_i) passes while actually carrying the package. According to characterizations (ii) and (iii), when defining the recursion, we have to take care of these two cases, see Figure 2:

a) Agent i might arrive at node v^* 'late', the package has already been dropped off there before by one of the agents $1, 2, \ldots, i - 1$ and had been waiting.

b) Agent i might arrive at node v^* 'early', in which case it should walk towards the package to receive it earlier and bring it back to v^* faster (having larger velocity than the currently carrying agent, after all). In this case, agent i picks up the package at a point p which is strictly in the interior of the edge $\{u, v^*\}$ and which is as close to node v^* as possible, i.e., p must be reachable by both agent i *and* the package – carried by only the first $i - 1$ agents – at the *earliest* possible time: $(d(p_i, v^*) + d(v^*, p))/v_i$.

Dynamic Program. First we are interested in the distance between any two nodes in the graph, which we can find with an *all-pair-shortest-paths* algorithm APSP. We denote the time needed for this precomputation by $\mathcal{O}(\text{APSP})$. Then, given the agents in ascending order of their velocities v_i, for each prefix $1, 2, \ldots i$ of the agent order and each node v we define the following subproblem:

$$\begin{aligned}
S\,[i, v] &= \text{A fastest schedule to bring the package to node } v \text{ using agents } \{1, \ldots, i\}.\\
\mathcal{T}\,[i, v] &= \text{The time needed in } S\,[i, v] \text{ to deliver the package to } v.\\
A\,[i, v] &= \text{Index of the last agent to carry the package in } S\,[i, v].\\
p\,[i, (u, v)] &= \text{The pick-up point } p \text{ strictly inside edge } \{u, v\} \text{ and closest to } v, \text{ reachable}\\
&\quad\ \text{by } \textit{both} \text{ the package (coming from } u, \text{ delivered by agents } 1, \ldots, i - 1)\\
&\quad\ \textit{and} \text{ agent } i \text{ (coming via } v) \text{ in time } (d(p_i, v) + d(v, p))/v_i \text{ (if applicable).}
\end{aligned}$$

Note that although our graph only has undirected edges, $p[i, (u, v)]$ considers an *ordered* tuple of nodes (u, v), denoting that the package is transported from u to v. Thus $p[i, (v, u)]$

has the analogous meaning of the package crossing edge $\{u,v\}$ from v towards u. Both $p[i,(u,v)]$ and $p[i,(v,u)]$ might be undefined, as can be seen below.

We compute the optimum delivery times $\mathcal{T}[i,v]$ (together with $A[i,v]$) without explicitly maintaining the schedules $S[i,v]$. A concrete final schedule S can then be retraced from $A[,]$, see Theorem 4. For computing $\mathcal{T}[i,v]$ and $A[i,v]$ we 'guess' the first node v^* of cases a) and b) above by trying each node v as a candidate. We then can compute $\mathcal{T}[i,v]$ and $A[i,v]$ for all other nodes using the pre-computed distances between all pair of nodes:

1. *Initialization:* For all nodes v, we initialize $S[i,v] := S[i-1,v], A[i,v] := A[i-1,v]$ and $\mathcal{T}[i,v] := \mathcal{T}[i-1,v]$. This automatically takes care of case a), where the package arrives at v before agent i can reach v.
2. *In-edge pick-ups:* We go over all node pairs (u,v) such that $\{u,v\} \in E$ and check whether agent i can pick up the package inside $\{u,v\}$ to advance it to node v faster than in schedule $S[i-1,v]$. We do so by checking whether we have $d(p_i,v)/\upsilon_i < \mathcal{T}[i-1,v]$ *and* $d(p_i,u)/\upsilon_i > \mathcal{T}[i-1,u]$. In this case, agent i receives the package from a previous agent j that brought it from u or from $p[j,(u,v)]$. Thus we get a set P of candidates for $p[i,(u,v)] := \arg\min_{p \in P} \{d(p,v)\}$. The candidate set P consists of all points p strictly inside the edge $\{u,v\}$ such that there exists an agent of index j, $A[i-1,u] \le j < i$, for which we have
$$\max\left\{\mathcal{T}[i-1,u], \frac{d(p_j,u)}{\upsilon_j}\right\} + \frac{d(u,p)}{\upsilon_j} = \frac{d(p_i,v)\ +\ d(v,p)}{\upsilon_i}$$
if j is coming from u, *or* – if $p[j,(u,v)]$ is defined –
$$\frac{d(p_j,v)\ +\ d(v,p[j,(u,v)])\ +\ d(p[j,(u,v)],p)}{\upsilon_j} = \frac{d(p_i,v)\ +\ d(v,p)}{\upsilon_i}.$$
Having computed $p[i,(u,v)]$ as the point in P closest to v, we update node v accordingly: Set $\mathcal{T}[i,v] := \min\left\{\mathcal{T}[i,v], \frac{d(p_i,v)+2d(p[i,(u,v)],v)}{\upsilon_i}\right\}$, where using 'min' takes care of cases in which we have multiple incident edges to v that all potentially have in-edge pick-ups by i, and set $A[i,v] = i$ (valid since we consider the case where $d(p_i,v) < \mathcal{T}[i-1,v]$).
3. *Updates:* So far we have computed the subproblems $S[i,v]$ correctly, if node v corresponds to the first node v^* of cases a) and b) (in particular we checked whether the faster agent i can help to advance the package over only one edge). Now we also consider all cases where agent i transports the package over arbitrary distances, by updating all other schedules $S[i,u]$ accordingly: For each node v, for each node u, if $\mathcal{T}[i,u] > \max\{\mathcal{T}[i,v], d(p_i,v)/\upsilon_i\} + d(v,u)/\upsilon_i$ we set $A[i,u] := i$ and $\mathcal{T}[i,u] := \max\{\mathcal{T}[i,v], d(p_i,v)/\upsilon_i\} + d(v,u)/\upsilon_i$.

▶ **Theorem 4.** *An optimum schedule for* FastDelivery *of a single package can be computed in time* $\mathcal{O}(k^2|E| + k|V|^2 + \text{APSP}) \subseteq \mathcal{O}(k^2n^2 + n^3)$.

Proof. For each i from 1 to k we can compute all values $A[i,v]$, $\mathcal{T}[i,v]$ in time $\mathcal{O}(|V|)$ for the initialization, $\mathcal{O}(|E|k)$ to check for in-edge pick-ups and $\mathcal{O}(|V|^2)$ for the updates (for which we need precomputed all-pair shortest paths). Overall we get a running time of $\mathcal{O}(\text{APSP} + k^2|E| + k|V|^2)$. The delivery time is then given in $\mathcal{T}[k,t]$. Correctness of the algorithm follows from the definition of the subproblems and the case distinction stemming from Lemma 2. Since we did not explicitly maintain the schedules $S[i,v]$, we retrace the concrete schedule S from $A[,]$ by backtracking: Let i denote the last used agent $A[k,t]$. We can find i's 'first node' v^* in time $\mathcal{O}(|V|)$ by searching for the smallest value $\mathcal{T}[i,u]$ such that
$$\max\{\mathcal{T}[i,u], d(p_i,u)/\upsilon_i\} = \mathcal{T}[k,t] - d(u,t)/\upsilon_i.$$

If $A[i, v^*] \neq i$, we recurse, otherwise we find the correct adjacent node and all in-edge handovers by looking – for each of the $\mathcal{O}(\deg(v^*))$ many neighbors u of v^* – at the overall $\mathcal{O}(k \deg(v^*))$ many values $p[j,(u,v^*)]$ (where $j \leq i$) and $\mathcal{T}[j,u]$ (where $j < i$). ◀

4 Prioritizing delivery time over energy consumption

In this Section, we want to find the most efficient among all fastest delivery schedules. We call this problem FastEfficientDelivery and will first show that it can be solved in polynomial time for uniform velocities ($\forall i, j\colon v_i = v_j$), due to a characterization of optimum schedules. In contrast, we prove NP-hardness for arbitrary speeds, even on planar graphs. However, for paths we show how one can subdivide general instances into phases of concecutive agents having the same velocity, and achieve an efficient $\mathcal{O}(n + k \log k)$-time algorithm.

4.1 A polynomial-time algorithm for uniform velocities

▶ **Lemma 5.** *Consider* FastEfficientDelivery *on instances with uniform agent velocities and let δ denote the offset of the closest agent's starting position to s. Then there exists an optimum schedule such that the pick-up position q_i^+ of each involved agent i satisfies:*
- *$d(s, q_i^+) + d(q_i^+, t) = d(s, t)$, i.e., q_i^+ lies on a shortest s-t-path, and*
- *$d(p_i, q_i^+) \leq \delta + d(s, q_i^+)$, with equality if q_i^+ lies strictly inside an edge.*

Proof. Since all agents have the same velocity $\overline{v}$, any fastest delivery of the package must follow a shortest path from s to t. Furthermore, since the closest agent could deliver the package on its own in time $(\delta + d(s,t))/\overline{v}$, each involved agent i has to arrive at q_i^+ no later than the package itself, giving $d(p_i, q_i^+) \leq \delta + d(s, q_i^+)$. It remains to show that we can modify every optimum solution into an optimum solution in which we have $d(p_i, q_i^+) = \delta + d(s, q_i^+)$ whenever q_i^+ lies strictly inside an edge $e = \{u, v\}$. Denote by i the first agent for which this is not the case and by $i-1$ its preceding agent. Assume that the package enters e via u (i.e. $d(s,u) < d(s,v)$). Note that i must have entered e via v, since otherwise the energy consumption could be improved by letting i pick up the package already at u (without increasing the delivery time), contradicting the optimality of our solution. Now we distinguish two cases relating the weights ω_i and ω_{i-1}, yielding either a decrease of the energy consumption, or a possibility to move q_i^+ to a position satisfying the characterization.
- $2\omega_i > \omega_{i-1}$: Moving q_i^+ by $\epsilon > 0$ towards v decreases $\mathcal{E}$ by an amount of $(2\omega_i - \omega_{i-1}) \cdot \epsilon > 0$.
- $2\omega_i \leq \omega_{i-1}$: We move q_i^+ towards u (without increasing neither delivery time nor energy consumption) until we reach $q_i^+ = u$, or q_i^+ inside the edge $\{u, v\}$ such that $d(p_i, q_i^+) = \delta + d(s, q_i^+)$, or $q_i^+ = q_{i-1}^+$. In the last case, discarding agent $i-1$ from our solution results in an energy consumption decrease of at least $\omega_{i-1} \cdot d(p_{i-1}, q_{i-1}^+) > 0$. ◀

Polynomial-time algorithm. We use the characterization in Lemma 5 to find an optimum solution for FastEfficientDelivery of delivery time $\mathcal{T} = (\delta + d(s,t))/\overline{v}$: For each agent i, we compute the set Q_i of all potential pick-up locations, i.e., the set of points q_i that satisfy Lemma 5. The number of potential locations is $|Q_i| \in \mathcal{O}(|V| + |E|) \subseteq \mathcal{O}(n^2)$. Then we build an auxiliary directed acyclic multi-graph H on a node set $V(H) = \bigcup_{i=1}^{k} Q_i$, of size $|V(H)| \in \mathcal{O}(|V| + k|E|) \subseteq \mathcal{O}(kn^2)$. Each directed edge in $E(H)$ describes how agent i can contribute to the delivery by bringing the package from its starting position q_i to another agent's starting position q_j along a shortest s-t-path: For each pair of nodes $q_i \in Q_i$ and $q_j \in V(H)$ such that $q_i \neq q_j$ and $d(s, q_i) + d(q_i, q_j) + d(q_j, t) = d(s, t)$, we add an arc (q_i, q_j) of

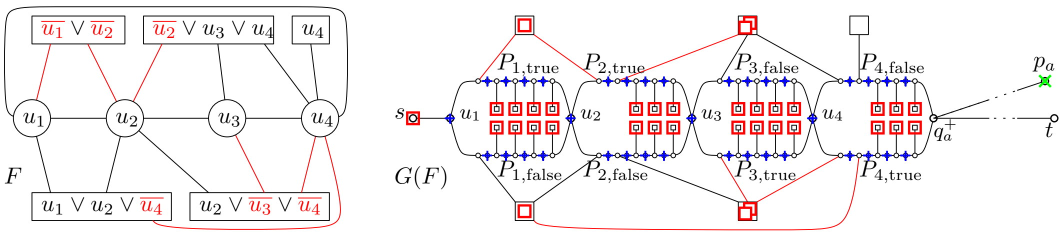

Figure 3 (left) A planar 3CNF formula F, satisfiable by $(u_1, u_2, u_3, u_4) = (\text{true, false, false, true})$. (right) Its transformation into a corresponding delivery graph $G(F)$. The satisfiable assignment of F corresponds to a low-cost delivery in $G(F)$ via paths $P_{1,\text{true}}, P_{2,\text{false}}, P_{3,\text{false}}, P_{4,\text{true}}$, and vice versa. We have slow agents for clauses ($\square$), fast agents for variables/literals ($+$) and a very fast agent ($\times$).

weight $\omega_i \cdot (d(p_i, q_i) + d(q_i, q_j))$ to $E(H)$. Overall, we have at most $|E(H)| \in \mathcal{O}(k \cdot n^2 \cdot kn^2)$ many arcs. By construction of H, running Dijkstra's shortest path algorithm on the multi-graph H finds a shortest path from s to t corresponding to an optimal solution.

▶ **Theorem 6.** *An optimum solution for* FastEfficientDelivery *can be found in time* $\mathcal{O}(k^2 n^4)$*, assuming all agents have the same velocity.*

4.2 NP-hardness on planar graphs

Contrary to FastDelivery (where we had non-decreasing velocities υ_i), when prioritizing delivery time *but still regarding energy consumption*, we can't characterize the order of the agents by their coefficients (υ_i, ω_i): Consider an instance in which both the starting position p_a of the absolutely fastest agent a as well as the package destination t are separated from the rest of the graph by two very long edges q_a^+—p_a, q_a^+—t. Then in *every* fastest solution, agent a (with υ_a large, e.g. 8) must deliver the package from q_a^+ to t, see Figure 3 (right).

In FastEfficientDelivery, the task is thus to balance slow but efficient agents (with, e.g., $\upsilon = 1, \omega = 0$) and fast inefficient agents (with, e.g., $\upsilon = 2, \omega = 1$) to collectively deliver the package to a's pick-up location q_a^+ *just-in-time* – i.e., in time $d(p_a, q_a^+)/\upsilon_a$ – *without using too much energy.* We can construct suitable instances by a reduction from Planar3SAT [23] *(Sketch)*: Starting from a planar formula F in three-conjunctive normal form, as in Figure 3 (left), we build a delivery graph $G(F)$. This can be done such that the instance is guaranteed to have schedules with minimum delivery time, i.e. with $\mathcal{T} = d(p_a, t)/\upsilon_a$. However, there should only be such a minimum-time schedule which simultaneously has low energy consumption $\mathcal{E}$ *if and only if* the formula F has a satisfiable variable assignment.

To this end, we place the fast agents on nodes corresponding to variables and literals. Intuitively, these agents decide on the routing of the package, thus setting the assignment of each variable. The slow agents, on the other hand, are placed on clause nodes, each clause receiving just one agent short of the number of its literals. Intuitively, for a just-in-time delivery to q_a^+ with small energy consumption, each clause has to spend one of its agents for each of its unsatisfied literals. By construction, this is only possible if each clause is satisfied:

▶ **Theorem 7.** FastEfficientDelivery *is* NP*-hard, even on planar graphs.*

4.3 An efficient algorithm for paths

The preceding hardness result raises the question for which restricted graph classes we can expect an efficient algorithm for arbitrary velocity instances. To contribute to this question it is natural to study paths – on paths, the V-shaped p_a—q_a—t component attached to the

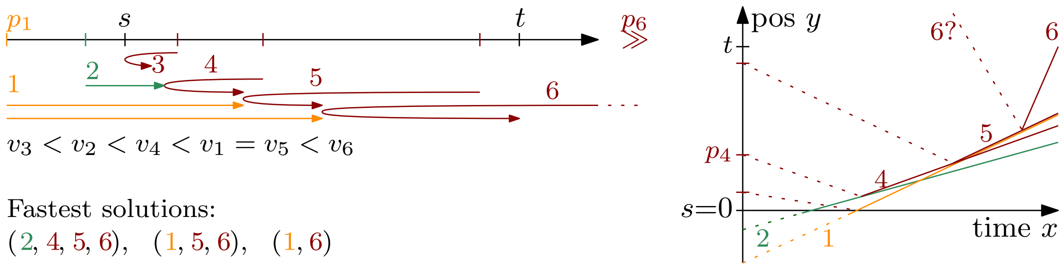

Figure 4 (left) Possible optima: If agent 4 is involved, it must take over the package from agent 2, since agent 3 is too slow. Using agents 2 and 4 to bring the package to agent 5's pick-up position takes the same time as using agent 1 on its own. Agents 1 and 5 have the same velocity, so in terms of delivery time we could use either or even both of them, but agent 1 only if agents 2 and 4 are both not used (otherwise they all consume energy). (right) Fastest solutions correspond to at most 1 agent with $p_i < s$ and a number of agents corresponding to a suffix of the upper envelope.

rest of the graph, as used in the hardness proof, 'collapses' to a line. We show that this allows us to decompose the problem into linearly many uniform velocity instances in time $\mathcal{O}(n + k \log k)$. Theorem 6 then implies that FastEfficientDelivery can be solved in polynomial-time. Improving on this by a careful analysis of paths, we show how to solve each uniform velocity instance in time $\mathcal{O}(n + k \log k)$ as well, and that these instances can be combined in time $\mathcal{O}(k)$, giving an overall $\mathcal{O}(n + k \log k)$-time algorithm.

Decomposition into uniform velocity instances

In the following, we look at the path graph G as the real line, and assume (after performing a depth-first search from s and ordering the starting positions in time $\mathcal{O}(n + k \log k)$) without loss of generality that $s = 0 < t$, that $p_1 \leq p_2 \leq \ldots \leq p_k$ and that $n = k + 2$, as the only relevant nodes on the line are s, t and the starting positions p_i. Note that in an optimum solution of FastEfficientDelivery, no agent i will ever take over the package from another agent j which i overtakes from the left. In particular, this means that we will need at most one agent with starting position $p_i < s$, and that after the package is picked up at s, it will never have to wait between a drop-off by an agent j and a pick-up by the next agent i, since j could continue carrying the package towards i, thus decreasing the overall delivery time. Hence in an optimum schedule we also have for consecutive agents i, j with $s < p_j < p_i$, that $v_j \leq v_i$ (otherwise we can discard i, by this decreasing the delivery time).

Decomposition. Assume that agent i is the agent that delivers the package to t. We represent the trajectory of the package while being carried by i as a ray giving the position y on the real line as a function $f_i(x)$ of the time x passed so far, see Figure 4 (right). We now inductively compute a set containing all functions $f_0, f_1, \ldots, f_k$, where $f_0(x) = s = 0$.

If we have $p_i < 0$, then by the reasoning above, i is the only involved agent, and the function is simply $f_i \colon y = v_i \cdot x + p_i$. For $p_i > s$, the slope v_i of the ray is set, but not its pick-up position. In order to minimize the earliest possible delivery time x (i.e. $f_i(x) = t$), by the non-decreasing velocity property i must pick up the package as early as possible – e.g. in Figure 4 (left), the fastest agent 6 would not get the package from agent 4, but from agent 5 who is able to speed up the transport between agents 4 and 6, thus advancing the last handover position and allowing agent 6 to pick up the package earlier.

Formally, the pick-up position is given by the time-wise first (or in other words leftmost) intersection of a query line $y = p_i - v_i \cdot x$ (modelling the agent moving towards s) with any preceding ray $f_0, \ldots, f_{i-1}$. Let $q_i := (x_i, y_i)$ denote the intersection point of the query line with the upper envelope of the preceding rays, and denote by f_j a ray of steepest slope υ_j among all rays $f_0, \dots, f_{i-1}$ that contain q_i – e.g. the query line "6?" in Figure 4 (right) intersects both f_1 and f_5 on the upper envelope, and since both have the same slope, we can consider either.

In case $\upsilon_j > \upsilon_i$, agent i will not be used in an optimal schedule and we set $f_i = 0$. If, however, $\upsilon_j \leq \upsilon_i$, then f_i is given by the line equality $f_i \colon y = \upsilon_i \cdot x + (y_i - \upsilon_i \cdot x_i)$. After completion, an optimum schedule corresponds to a path along the rays of our diagram from $(0, 0)$ to the ray reaching $y = t$ at the earliest possible time.

Fast computation and recombination. To quickly compute the equation of each ray f_i, we need to find the intersection of a query line with the upper envelope of $\mathcal{O}(k)$ many rays. Precomputing this envelope as an ordered list of its segments would allow us to speed up the intersection queries from a linear to a binary search (*convex hull trick for dynamic programming* [21]). However, the set of functions that we query here is not known up front. Instead, we apply the classic geometric point-line duality [18]. In this dual setting, the task of finding the leftmost intersection point of a query line with a set of lines turns into finding a right tangent from a query point (the dual of the query line) onto the convex hull of a point set (the dual of the rays f_i). The dynamic planar convex hull data structure by Brodal and Jacob [4, 22] allows point insertions and tangent queries all in $\mathcal{O}(\log k)$ amortized time, giving an overall running time of $\mathcal{O}(k \log k)$. Assuming that we know the optimum schedule for each of the uniform velocity intervals, it remains to recombine these subschedules:

▶ **Lemma 8.** *Arbitrary velocity instances of* FastEfficientDelivery *on paths can be decomposed into and recombined from uniform velocity instances in time $\mathcal{O}(n + k \log k)$.*

A fast algorithm for uniform velocity instances on the line

We are left to solve the case where all agents have the same uniform velocity $\overline{\upsilon}$. As before, we denote by δ the offset of the closest agent's starting position to s, and let a denote the corresponding agent. No agent i with $p_i < s$ other than maybe agent a is involved in an optimum schedule (all others would only slow down delivery). Also note that if $p_a < s$, the setting is equivalent to one where a starts at $s + (s - p_a)$, so we can assume (after relabelling the agents) $a = 1$, $\delta = p_1$. This also implies $\mathcal{T} = (\delta + (t - s))/\overline{\upsilon}$ and we can ignore agents i that are dominated by earlier, cheaper agents j with $p_j < p_i$ and $\omega_j < \omega_i$.

Towards a dynamic program. We define the point q_i as the leftmost point on the line where agent i can pick up the package without causing a delay, i.e., we have $q_i := \frac{p_i + s - \delta}{2}$ since $p_i - q_i = \delta + (q_i - s)$. Note that $q_1 = s$ and $q_j < q_i$ for $j < i$. Similarly as – but more specific than – in the characterization of uniform instances on general graphs (Lemma 5) we get a limited set of possible pick-up locations:

▶ **Lemma 9.** *There is an optimum solution where each agent i that is involved in advancing the package picks it up at $q_i^+ = q_i$ or at $q_i^+ = p_i$.*

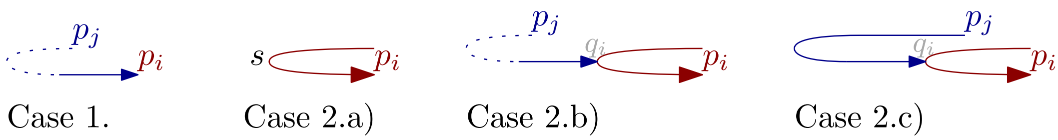

Figure 5 Case distinction in the dynamic program for FASTEFFICIENTDELIVERY on the line. Either agent i is not involved at all, does all on its own or is subsequent to some agent j, where we distinguish between $p_j \le q_i$ and $p_j > q_i$.

Dynamic program. Lemma 9 suggests that in an inductive approach from left to right it suffices to consider only finitely many handover options. We define the following subproblems:

$$\begin{aligned} S[i] &= \text{An energy-optimal schedule to deliver the package to } p_i \\ &\quad\ \text{in time } (\delta + (p_i - s))/\overline{v}, \text{ using only the agents } \{1, 2, \ldots, i\}. \\ \mathcal{E}[i] &= \text{Energy consumption of } S[i]. \\ A[i] &= \text{Index of the last package-carrying agent in } S[i]. \\ A'[i] &= \text{Index of the second to last carrying agent in } S[i] \text{ (if any)}. \end{aligned}$$

We will argue how to compute the optimum energy costs $\mathcal{E}[i]$ (and with it $A[i]$ and $A'[i]$) without explicitly maintaining the schedules $S[i]$ ($S[i]$ can later be retraced from $A[i]$ and $A'[i]$). For computing $\mathcal{E}[i]$, $A[i]$ and $A'[i]$, we distinguish four cases (also shown in Figure 5):

1. Agent i is not involved in $S[i]$.
2. Agent i is involved in $S[i]$. Hence by Lemma 9, agent i has pick-up location $q_i^+ = q_i$; and we get the following three variations:
   a. $i = 1$ and agent 1 picks up the package at s itself.
   b. Agent i picks up the package from some other agent j with $p_j \le q_i$.
   c. Agent i picks up the package from some other agent j with $p_j > q_i$.

In cases 1, 2b) and 2c), we can determine $\mathcal{E}[i]$ in constant time using a single prior entry of the dynamic programming table:

Case 1. If i is not involved in $S[i]$, the best choice for the agent who transports the package to p_i is agent $i-1$, as it is the cheapest one on the last segment $[p_{i-1}, p_i]$ and we have $\mathcal{E}[i-1] \le \mathcal{E}[j] + (p_{i-1} - p_j) \cdot \omega_j$ for all $j < i-1$ by induction. Hence we can optimize in constant time:

$$\begin{aligned} \mathcal{E}[i] &= \min_{j<i}\{\mathcal{E}[j] + (p_i - p_j) \cdot \omega_j\} = \mathcal{E}[i-1] + (p_i - p_{i-1}) \cdot \omega_{i-1}, \\ A[i] &= i - 1 \text{ and } A'[i] = A[i-1]. \end{aligned}$$

Case 2.a) This is the base case where the first agent is on its own:

$$\mathcal{E}[1] = 2 \cdot |p_1 - s| \cdot \omega_1 = 2 \cdot \delta \cdot \omega_1, \quad A[1] = \text{none}, \ A'[1] = \text{none}.$$

Case 2.b) If agent i is involved in $S[i]$ and takes over at q_i from an agent j with $p_j \le q_i$, we want j to minimize $\mathcal{E}[j] + (q_i - p_j) \cdot \omega_j$, the cost of bringing the package to q_i. Now let $i' = \max\{j \mid p_j \le q_i\}$ be the agent starting closest to the left of q_i. As in Case 1, we argue that i' is the optimum choice for j, as it minimizes the cost on $[p_{i'}, q_i]$ and does not constrain the schedule up to $p_{i'}$ further. Hence, we again get in amortized constant time, i.e., we update i' by incrementing it lazily when going from i to $i+1$:

$$\mathcal{E}[i] = \mathcal{E}[i'] + (q_i - p_{i'}) \cdot \omega_{i'} + 2 \cdot (p_i - q_i) \cdot \omega_i, \quad A[i] = i, \ A'[i] = i'.$$

The most interesting case is the remaining case (2.c), where the agent j handing over to i starts in between q_i and p_i. Where can we look up the energy consumption c of an optimum schedule that ends with j bringing the package q_i – the dynamic program being only defined for points p_j? For some j, we might have $A[j] = j$, so $S[j]$ ends by j walking to q_j and back. In that case, we can exploit $q_j < q_i$ and use $\mathcal{E}[j] - (p_j - q_i) \cdot \omega_j$ as a candidate for the energy consumption c. But what if $A[j] \neq j$? As we saw, this implies $A[j] = j - 1$, but in that case we cannot just subtract $(p_j - q_i)\omega_j$: We do not know how $S[j]$ looks like between q_i and p_j. We argue that we do not need to consider these agents j as candidates at all!

▶ **Lemma 10.** *If in some optimal schedule $S[i]$ the agent j preceding i is of type 2.c), then in the schedule $S[j]$ we have $A[j] = j$.*

Proof. Under the assumption of Case 2.c), the cost of agent i is fixed to $2 \cdot (p_i - q_i) \cdot \omega_i$. Agents 1 to $i - 1$ will collaborate in the most efficient way to bring the package up to q_i. By definition of j, j is the last agent bringing the package to q_i. From the decreasing weight property, we know that none of the agents $j + 1$ to $i - 1$ were involved in $S[i]$. So if we take the partial schedule of $S[i]$ up to q_i and extend it by letting j bring the package to p_j, we obtain a feasible candidate schedule S' for $S[j]$ as none of the agents $j + 1$ to i are involved. We now argue that S' is an optimum schedule for $S[j]$. The segment $[q_i, p_j]$ is covered with the minimum possible energy, as j is the *unique* most efficient agent available for $S[j]$. The segment $[s, q_i]$ is also covered cheapest possible as its part of $S[i]$ was optimized over all agents 1 to i, so a superset of the agents available for $S[j]$. Moreover, the uniqueness implies that *all* optimum schedules for $S[j]$ need to end with agent j on $[q_i, p_j]$, hence $A[j] = j$. ◀

Case 2.c) Lemma 10 leaves us with only those agents j whose schedules $S[j]$ we understand sufficiently to modify them into candidates for $S[i]$ under Case 2.c):

$$\mathcal{E}[i] = \min_j \{\mathcal{E}[j] - (p_j - q_i)\omega_j \mid q_i < p_j < p_i \wedge A[j] = j\} + 2(p_i - q_i)\omega_i,$$
$$A[i] = i, \quad A'[i] = \arg\min_j \{\mathcal{E}[j] - (p_j - q_i)\omega_j \mid q_i < p_j < p_i \wedge A[j] = j\}.$$

We can now take $\mathcal{E}[i]$ as the minimum over the four cases 1–2.c) and compute all schedules $S[i]$ by proceeding over all subproblems in increasing order, giving us the energy-optimal schedules for delivering the package to the points p_i in time $(\delta + (p_i - s))/\overline{v}$. How can we use the solutions to the subproblems $\mathcal{E}[i]$ to find the energy $\mathcal{E}$ of an energy-optimal schedule delivering the package to the target t in optimum time $(\delta + (t - s))/\overline{v}$? Let k' be the closest agent on the left of t, i.e., $k' := \arg\max_i p_i \leq t$. Clearly, if in an optimum schedule the package is delivered to t by an agent starting to the left of t, then by the decreasing weight property this agent must be agent k', giving us $\mathcal{E} = \mathcal{E}[k'] + (t - p_{k'}) \cdot \omega_k$.

Delivery to t and agents with $p_i > t$. It remains to take care of agents with starting positions $p_i > t$: As illustrated in Figure 6, multiple agents with $p_i > t$ might be involved in the most efficient delivery. Note that our dynamic programming problem $\mathcal{E}[i]$ is defined independent of t and so we can also easily compute $\mathcal{E}[i]$ for $p_i > t$. Agents i with $q_i > t$ are not useful, however, for a delivery to t, as they arrive in $[s, t]$ only after the package has been delivered. Similar to Lemma 10, we claim that among the remaining agents i only those with $A[i] = i$ need to be considered:

▶ **Lemma 11.** *If an agent i with $p_i > t$ is the last agent in any optimal schedule S from s to t, then $A[i] = i$.*

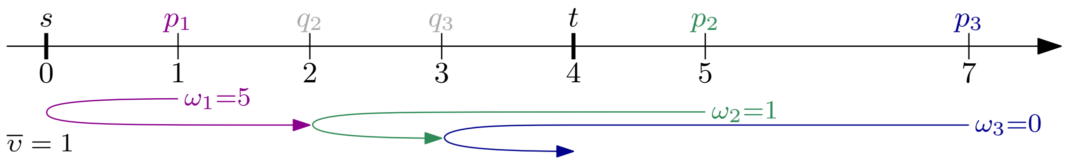

Figure 6 An example, where the only optimum schedule uses both agents on the right of t. The optimum schedule has delivery time $\mathcal{T} = (p_1 + t - s)/\overline{v} = 5/1 = 5$ and energy consumption $\mathcal{E} = p_1 \cdot \omega_1 + (p_2 - q_2 + q_3 - q_2) \cdot \omega_2 + (p_3 - q_3 + t - q_3) \cdot \omega_3 = 3 \cdot 5 + 4 \cdot 1 + 5 \cdot 0 = 19$.

Proof. We have $q_i < t < p_i$. By the decreasing weight property, no agent $j > i$ will be used in S. We extend S to a schedule S' by letting agent i walk from t to p_i. Then S' is a candidate for $S[i]$. Similar to Lemma 10, S' consists of an optimal solution for $[s, t]$ and the strictly cheapest agent on $[t, p_i]$ and hence S' is optimal for $S[i]$ and all optimum schedules for $S[i]$ have $A[i] = i$. The optimum s-t-delivery is thus given by:

$$\mathcal{E} = \min_j \{\mathcal{E}[j] - (p_j - t)\omega_j \mid (q_j < t < p_j \text{ and } A[j] = j) \text{ or } j = k'\},$$

which takes linear time once at the very end. ◀

Details of the dynamic program. The computational bottleneck of our dynamic program is (for each subproblem $\mathcal{E}[i]$) the minimization over the set of options in Case 2.c). Each option evaluates a linear function $f_j(q_i) := \omega_j \cdot q_i + (\mathcal{E}[j] - p_j \cdot \omega_j)$ at position q_i, which can be seen as a lower envelope intersection query. Similarly to before, we use point-line duality and a dynamic convex hull data structure to avoid considering all agents explicitly as predecessors and instead quickly search the best one.

▶ **Lemma 12.** *An optimum schedule for* FastEfficientDelivery *with uniform velocity $\overline{v}$ on the line can be computed in $\mathcal{O}(n + k \log k)$ time.*

Combining this with Lemma 8 gives the full solution on paths. Note that strictly speaking, in the uniform velocity instances, the package is not available at s at time zero, but is brought there by agents of preceding instances at exactly the time when the first agent can reach it.

▶ **Theorem 13.** *An optimum solution for* FastEfficientDelivery *on paths can be computed in $\mathcal{O}(n + k \log k)$ time.*

5 Optimizing convex combinations of objectives

In this section, we look at a convex combination of the two objectives: minimizing both the delivery time $\mathcal{T}$ and the energy consumption $\mathcal{E}$ by minimizing the term $\epsilon \cdot \mathcal{T} + (1 - \epsilon) \cdot \mathcal{E}$, for a given value ϵ, $0 < \epsilon < 1$. We call the problem of minimizing this combined objective CombinedDelivery. As an application of the NP-hardness proof for FastEfficientDelivery, we get NP-hardness of CombinedDelivery as well: The main idea is to counter small values of ϵ by scaling the weights of the agents by a small factor $\delta(\epsilon)$, thus decreasing the importance of $\mathcal{E}$ alongside $\mathcal{T}$ as well.

▶ **Theorem 14.** CombinedDelivery *is* NP*-hard for all $\epsilon \in (0, 1)$, even on planar graphs.*

A 2-approximation for CombinedDelivery using a single agent

Recall that for FastDelivery, the agents involved in an optimum delivery were characterized by increasing velocities υ_i, while for FastEfficientDelivery on path graphs, the agents of an optimum solution were characterized by decreasing tuples $(\upsilon_i^{-1}, \omega_i)$.

Although it is *not* possible to characterize the *order* of the agents in an optimum CombinedDelivery schedule by their velocities and weights alone, we can at least characterize the position of a *minimal* agent, leading to a 2-approximation using a single agent:

▶ **Lemma 15.** *Let without loss of generality* $1, 2, \ldots, i$ *denote the indices of all involved agents appearing in that order in an optimum* CombinedDelivery *schedule. Then the last agent* i *is* minimal *in the following sense:* $i \in \arg\min_j \left\{\epsilon \cdot \upsilon_j^{-1} + (1-\epsilon) \cdot \omega_j\right\}$.

Proof. Recall that we denote by d_j the total distance travelled by agent j and by d_j^* the distance travelled by agent j while carrying the package. Thus agent j contributes at least $\epsilon \cdot d_j^* \cdot \upsilon_j^{-1} + (1-\epsilon) \cdot (d_j \cdot \omega_j) \geq d_j^* \cdot \left(\epsilon\upsilon_j^{-1} + (1-\epsilon)\omega_j\right)$ towards $\epsilon\mathcal{T} + (1-\epsilon)\mathcal{E}$. Assume for the sake of contradiction that the minimum value $\epsilon\upsilon_j^{-1} + (1-\epsilon)\omega_j$ is not obtained by agent i but by an agent $m < i$. Then we can replace the agents $m+1, \ldots, i$ by agent m, resulting in a decrease in the objective function of at least

$$
\begin{aligned}
&\sum_{j=m}^{i} \epsilon d_j^* \upsilon_j^{-1} + (1-\epsilon) d_j \omega_j - \sum_{j=m}^{i} d_j^* \left(\epsilon\upsilon_m^{-1} + (1-\epsilon)\omega_m\right) \\
&\geq \sum_{j=m}^{i} d_j^* \left(\epsilon\upsilon_j^{-1} + (1-\epsilon)\omega_j\right) - \sum_{j=m}^{i} d_j^* \left(\epsilon\upsilon_m^{-1} + (1-\epsilon)\omega_m\right) \\
&\geq d_i^* \left(\epsilon\upsilon_i^{-1} + (1-\epsilon)\omega_i\right) - d_i^* \left(\epsilon\upsilon_m^{-1} + (1-\epsilon)\omega_m\right) > 0,
\end{aligned}
$$

contradicting the minimality of the optimum CombinedDelivery schedule. ◀

▶ **Theorem 16.** *There is a 2-approximation for* CombinedDelivery *which uses only a single agent (and thus can be found in polynomial time).*

Proof. Note that agent i contributes at most $\epsilon d_i \upsilon_i^{-1} + (1-\epsilon) d_i \omega_i$ towards $\epsilon\mathcal{T} + (1-\epsilon)\mathcal{E}$. Starting from an optimum CombinedDelivery schedule we can replace all agents $1, \ldots, i-1$ along their trajectories by the minimal agent i. This prolongs the travel distance of agent i by $2 \cdot \sum_{j=1}^{i-1} d_j^*$. Overall, we increase the objective function by at most

$$
\begin{aligned}
&2\sum_{j=1}^{i-1} d_j^* \left(\epsilon\upsilon_i^{-1} + (1-\epsilon)\omega_i\right) - \sum_{j=1}^{i-1} d_j^* \left(\epsilon\upsilon_j^{-1} + (1-\epsilon)\omega_j\right) \leq \sum_{j=1}^{i-1} d_j^* \left(\epsilon\upsilon_i^{-1} + (1-\epsilon)\omega_i\right) \\
&\leq \sum_{j=1}^{i-1} d_j^* \left(\epsilon\upsilon_j^{-1} + (1-\epsilon)\omega_j\right) < \sum_{j=1}^{i} d_j^* \left(\epsilon\upsilon_j^{-1} + (1-\epsilon)\omega_j\right) \leq \epsilon\mathcal{T} + (1-\epsilon)\mathcal{E}.
\end{aligned}
$$

Hence only using agent i to deliver the package is a 2-approximation for CombinedDelivery. We get a polynomial-time approximation algorithm with approximation ratio 2 by choosing among all k agents the one with minimum value $\left(\epsilon\upsilon_j^{-1} + (1-\epsilon)\omega_j\right) \cdot (d(p_j, s) + d(s, t))$. ◀

6 Discussion

Our techniques and results extend to a variety of delivery problems and model generalizations. A key ingredient here is that the order of our dynamic programming subproblems depends only on the parameters the agent has *while carrying the package*. Hence it is possible to, e.g., incorporate 2-speed agent models (modeling different speeds [16] with/without carrying the package) or topographical features (modeling edge traversals in uphill/downhill direction).

Furthermore, the 2-approximation given for CombinedDelivery is applicable to FastDelivery as well, and a relaxation of FastEfficientDelivery, in which one allows the optimum delivery time to be achieved with a constant-factor approximation of the energy consumption, stays NP-hard. It is unclear whether FastEfficientDelivery can be *solved efficiently on trees* or whether CombinedDelivery *allows a PTAS*. We consider these two problems the major open questions raised by this work.

References

1 Julian Anaya, Jérémie Chalopin, Jurek Czyzowicz, Arnaud Labourel, Andrzej Pelc, and Yann Vaxès. Convergecast and Broadcast by Power-Aware Mobile Agents. *Algorithmica*, 74(1):117–155, 2016. See also DISC'12. doi:10.1007/s00453-014-9939-8.

2 David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, NJ, USA, 2007. ISBN: 978-0-691-12993-8.

3 Evangelos Bampas, Jurek Czyzowicz, Leszek Gasieniec, David Ilcinkas, Ralf Klasing, Tomasz Kociumaka, and Dominik Pająk. Linear Search by a Pair of Distinct-Speed Robots. In *23rd International Colloquium on Structural Information and Communication Complexity SIROCCO'16*, pages 195–211, 2016. doi:10.1007/978-3-319-48314-6_13.

4 Gerth Stølting Brodal and Riko Jacob. Dynamic Planar Convex Hull. In *43rd Symposium on Foundations of Computer Science FOCS'02*, pages 617–626, 2002. doi:10.1109/SFCS.2002.1181985.

5 Andreas Bärtschi. *Efficient Delivery with Mobile Agents*. PhD thesis, ETH Zürich, 2017. doi:10.3929/ethz-b-000232464.

6 Andreas Bärtschi, Jérémie Chalopin, Shantanu Das, Yann Disser, Barbara Geissmann, Daniel Graf, Arnaud Labourel, and Matúš Mihalák. Collaborative Delivery with Energy-Constrained Mobile Robots. In *23rd International Colloquium on Structural Information and Communication Complexity SIROCCO'16*, pages 258–274, 2016. doi:10.1007/978-3-319-48314-6_17.

7 Andreas Bärtschi, Jérémie Chalopin, Shantanu Das, Yann Disser, Barbara Geissmann, Daniel Graf, Arnaud Labourel, and Matúš Mihalák. Collaborative delivery with energy-constrained mobile robots. *Theoretical Computer Science*, 2017. To appear. See also SIROCCO'16. doi:10.1016/j.tcs.2017.04.018.

8 Andreas Bärtschi, Jérémie Chalopin, Shantanu Das, Yann Disser, Daniel Graf, Jan Hackfeld, and Paolo Penna. Energy-efficient Delivery by Heterogeneous Mobile Agents. In *34th International Symposium on Theoretical Aspects of Computer Science STACS'17*, pages 10:1–10:14, 2017. doi:10.4230/LIPIcs.STACS.2017.10.

9 Andreas Bärtschi, Daniel Graf, and Paolo Penna. Truthful Mechanisms for Delivery with Agents. In *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems ATMOS'17*, pages 2:1–2:17, 2017. doi:10.4230/OASIcs.ATMOS.2017.2.

10 Andreas Bärtschi and Thomas Tschager. Energy-Efficient Fast Delivery by Mobile Agents. In *21st International Symposium on Fundamentals of Computation Theory FCT'2017*, pages 82–95, 2017. doi:10.1007/978-3-662-55751-8_8.

11 Jérémie Chalopin, Shantanu Das, Matús Mihalák, Paolo Penna, and Peter 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*, pages 111–122, 2013. doi:10.1007/978-3-642-45346-5_9.

12 Jérémie Chalopin, Riko Jacob, Matús Mihalák, and Peter Widmayer. Data Delivery by Energy-Constrained Mobile Agents on a Line. In *41st International Colloquium on Automata, Languages, and Programming ICALP'14*, pages 423–434, 2014. doi:10.1007/978-3-662-43951-7_36.

13 Jurek Czyzowicz, Krzysztof Diks, Jean Moussi, and Wojciech Rytter. Communication Problems for Mobile Agents Exchanging Energy. In *23rd International Colloquium on Structural Information and Communication Complexity SIROCCO'16*, 2016. doi:10.1007/978-3-319-48314-6_18.

14 Jurek Czyzowicz, Leszek Gasieniec, Konstantinos Georgiou, Evangelos Kranakis, and Fraser MacQuarrie. The Beachcombers' Problem: Walking and searching with mobile robots. *Theoretical Computer Science*, 608:201–218, 2015. doi:10.1016/j.tcs.2015.09.011.

15 Jurek Czyzowicz, Leszek Gasieniec, Adrian Kosowski, and Evangelos Kranakis. Boundary Patrolling by Mobile Agents with Distinct Maximal Speeds. In *19th European Symposium on Algorithms ESA'11*, pages 701–712, 2011. doi:10.1007/978-3-642-23719-5_59.

16 Jurek Czyzowicz, Konstantinos Georgiou, Evangelos Kranakis, Fraser MacQuarrie, and Dominik Pająk. Fence patrolling with two-speed robots. In *5th International Conference on Operations Research and Enterprise Systems ICORES'16*, pages 229–241, 2016. doi:10.5220/0005687102290241.

17 Erik D. Demaine, Mohammadtaghi Hajiaghayi, Hamid Mahini, Amin S. Sayedi-Roshkhar, Shayan Oveisgharan, and Morteza Zadimoghaddam. Minimizing movement. *ACM Transactions on Algorithms*, 5(3):1–30, 2009. See also SODA'07. doi:10.1145/1541885.1541891.

18 Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer, Heidelberg, Germany, 1987. doi:10.1007/978-3-642-61568-9.

19 Jack Edmonds and Ellis L. Johnson. Matching, euler tours and the chinese postman. *Mathematical Programming*, 5(1):88–124, 1973. doi:10.1007/BF01580113.

20 Greg N. Frederickson, Matthew S. Hecht, and Chul E. Kim. Approximation Algorithms for Some Routing Problems. *SIAM Journal on Computing*, 7(2):178–193, 1978. See also FOCS'76. doi:10.1137/0207017.

21 Woburn Collegiate Institute's Programming Enrichment Group. Convex hull trick. PEGWiki, September 2016. URL: http://wcipeg.com/wiki/Convex_hull_trick.

22 Riko Jacob. *Dynamic planar convex hull*. PhD thesis, Department of Computer Science, University of Aarhus, Denmark, 2002. BRICS Dissertation Series DS-02-3. URL: http://www.brics.dk/DS/Ref/BRICS-DS-Ref/BRICS-DS-Ref.html#BRICS-DS-02-3.

23 David Lichtenstein. Planar Formulae and Their Uses. *SIAM Journal on Computing*, 11(2):329–343, 1982. doi:10.1137/0211025.

24 Swiss Post. Swiss Post delivery robots in use by Jelmoli. Press release, August 2017. URL: https://www.post.ch/en/about-us/company/media/press-releases/2017/swiss-post-delivery-robots-in-use-by-jelmoli.

25 Paolo Toth and Daniele Vigo, editors. *The Vehicle Routing Problem*. SIAM Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. ISBN: 0-89871-498-2.

26 Elizabeth Weise. Amazon delivered its first customer package by drone. USA Today, December 2016. URL: http://usat.ly/2hNgf0y.