Data Delivery by Energy-Constrained Mobile Agents on a Line

Jérémie Chalopin[1], Riko Jacob[2], Matúš Mihalák[2], and Peter Widmayer[2]

[1] LIF, Aix-Marseille University & CNRS, Marseille, France
[2] Institute of Theoretical Computer Science, ETH Zurich, Zürich, Switzerland

Abstract. We consider n mobile agents of limited energy that are placed on a straight line and that need to collectively deliver a single piece of data from a given source point s to a given target point t on the line. Agents can move only as far as their batteries allow. They can hand over the data when they meet. In this paper we show that deciding whether the agents can deliver the data is (weakly) NP-complete, and for instances where all input values are integers, we present a quasi-, pseudo-polynomial time algorithm that runs in time $O(\Delta^2 \cdot n^{1+4\log \Delta})$, where Δ is the distance between s and t. This answers an open problem stated by Anaya et al. (DISC 2012).

Keywords: Mobile agents and robots; data aggregation and delivery; power-awareness; algorithms; complexity

1 Introduction

The production of inexpensive, simple-built, mobile robots has led to new research questions in how to employ and operate a *swarm* of such robots to achieve desired goals. One of the fundamental goals of robotics is the delivery of data from given sources to specified targets. An *energy-efficient* operation of mobile robots becomes crucial when batteries of the robots are limited.

In this paper we study how to efficiently operate robots (called *agents* in this paper) of limited batteries that need to collectively deliver one piece of data along a line from a *single* source to a *single* target. Formally, our setting is given by a *source* s and a *target* t placed along a line, and n autonomous mobile agents, where agent i, $i = 1, 2, \ldots, n$, has an initial *position* a_i and an initial *range* R_i, denoting the maximum length of a walk the agent can do. We ask whether the agents can *deliver* a *message* from source s to target t. The message is *picked up* by the first agent that reaches point s. An agent i with the message can *pass on* the message to agent j, if i and j meet at the same point on the line. The message is *delivered* if an agent with the message reaches target t. No agent $i = 1, \ldots, n$ can travel more than its range R_i. We refer to this problem as DataDelivery.

Obviously, it makes no sense for an agent to carry the message more than once. Then, even though the agents can in principle move simultaneously at a

time, it is easy to observe that for the sake of completing the task only,[3] the agents can move in turns: in the first turn, the agent picking up the message at s moves; then, the agent taking over from the first agent moves; then, the agent taking over from the second agent moves, and so on. In this view, a solution to DataDelivery can be given in form of a *schedule* that prescribes the subset of the agents that move and the order in which they move. We call a schedule which indeed delivers the message from s to t a *feasible schedule*.

Previously, this problem has been studied in edge-weighted graphs and with multiple sources [5]. Besides other results, it has been shown that the problem is NP-complete (for general graphs), and a $\min\{3, 1 + \max_{i,j} \frac{R_i}{R_j}\}$-*resource augmented* algorithm has been presented; here, a *polynomial-time* algorithm is called a γ-*resource augmented*, $\gamma > 1$, if either the algorithm (correctly) answers that there is no feasible schedule, or it finds a feasible schedule for the modified (augmented) powers $R'_i := \gamma \cdot R_i$. The complexity of the problem for the case when the graph is a line has been left open (it has been raised as an open question by Anaya et al. [3], but not studied).

In this paper, we close the open problem in that we show that DataDelivery is *weakly* NP-complete (even if all input values are integers), and at the same time, if all input values are integers, we present a quasi-, pseudo-polynomial time algorithm running in time $O\left(\Delta^2 \cdot n^{1+4\log \Delta}\right)$ and in time $O\left(\Delta^2 \cdot n^{1+4\log(R_{\max}+1)}\right)$, where Δ is the distance between s and t, and $R_{\max} := \max_i R_i$.

Related Work

There are very few papers studying explicitly the algorithmic question of data-aggregation-like problems by mobile agents with limited batteries. Besides the already mentioned paper by Chalopin et al. [5], the work of Anaya et al. [3] comes closest to our problem. Anaya et al. [3] study the *convergecast* problem: given a set of mobile agents in an edge-weighted graph, each agent possessing a certain piece of data, and having a uniform battery power B, the agents need to move, not more than what the battery allows, such that at some point at least one agent knows all data. Obviously, there are no fixed source and target nodes, which constitutes a difference to our problem. However, the main difference is that in [3], it is assumed that all agents have the same range. In this case, the problem is polynomial on a line, but is NP-hard if the graph is a tree.

Our problem has a flavour of the well studied problem of data aggregation in (wireless) sensor networks [8], where the general computational problem is to schedule the communication between (mostly) stationary sensor nodes so that all data collected by the individual sensors eventually arrive in a pre-specified aggregation node. While in data aggregation, the data is being sent over communication channels, in our setting, the agents physically deliver the data.

[3] It is an interesting, more general, and thus an even more difficult algorihtmic question to also minimize the time needed for the delivery; for this objective, parallel simultaneous moving of the agents would be crucial.

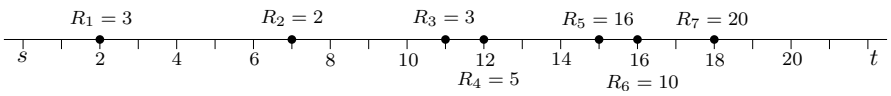

Fig. 1. A solvable instance of DataDelivery on a line with agents $a_1, a_2, \ldots, a_7$ depicted by the small full disks, and annotated by their respective ranges $R_1, R_2, \ldots, R_7$.

Power-aware computation with mobile agents is a relatively new research area, and, consequently, there is little algorithm-theoretical research. As an exception, Heo and Varshney [7] study self-deployment of agents in this context.

A related and intensively studied research question is that of minimizing the total travelled distance by agents (which have unlimited battery) [1], [2], [4].

Further Terminology, Notation, and Model Refinement

We consider the positions of s, t and all n agents to be given by their distance from s on the line. For simplicity, we identify the source s and the target t with this distance, i.e., we set $s = 0$ and we interpret $t > 0$ as the distance of the target from s. The position of agent i, $i = 1, \ldots, n$, is given by its distance a_i from s. Sometimes, we will refer to agent i by its position, i.e., we say agent a_i to denote agent i. Recall that R_i is the range of agent i, and let $R_{\max} = \max\{R_i \mid 1 \leq i \leq n\}$. Figure 1 gives a (solvable)[4] instance of DataDelivery on a line.

We will assume, without loss of generality, that all a_i are between s and t: if there is an agent a_i lying left of s, we can move it to s (and reduce its range correspondingly), and similary, we can move any agent a_j right of t to t (and reduce its range correspondingly). Obviously, the original instance has a solution if and only if the modified instance has one. In this adjusted problem instance, we may assume that $R_i < 2t$ for any agent i (as otherwise agent i with $R_i \geq 2t$ can deliver the message on its own).

Furthermore, we can assume that there is no agent at s. If there should be such an agent a_i, we use $s' = s + R_i$ as the new starting position. Now, any schedule from s to t can still be used (starting from the point in time when the packet passed position s') to deliver the data from s' to t, the only agent no longer available is a_i who cannot be used beyond s' anyway. Finally, we adjust all agents to the left of s' as described above. If this leads to an agent being positioned at s', we repeat the process.

2 The Quasi-, Pseudo-Polynomial Time Algorithm

In this section we present a dynamic-programming based algorithm for finding a feasible schedule, if the ranges and the positions of the agents are integers. We restrict ourselves to a specific class of feasible schedules: we call a feasible schedule $(a_{i_1}, a_{i_2}, \ldots, a_{i_j}, \ldots)$ *normalized*, if

[4] A solution is the schedule $(a_5, a_1, a_7, a_2, a_4, a_3, a_6)$.

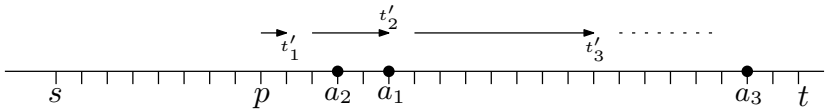

Fig. 2. Agents $a_1, a_2, a_3 \in A_p$ are p-crossing. The intervals (s'_i, t'_i) are lower bounds on the growth of the intervals (s_i, t_i): $s_i \geq s'_i$ and $t_i \geq t'_i$.

1. The positions where agents exchange the message are integers.
2. Every agent a_{i_j} walks with the message as far towards t as its range allows, with the exception when the agent reaches the initial position of the next agent in the feasible schedule, i.e., agent $a_{i_{j+1}}$.
3. The length of the schedule is minimal, i.e., we cannot remove any agent from the schedule and maintain its feasibility. This means, for example, that no agent a_{i_j} can reach (by exhausting its range) the point at which $a_{i_{j+2}}$ picks up the data from $a_{i_{j+1}}$ (making $a_{i_{j+1}}$ obsolete).

It is easy to see that in our "integer setting", if there exists a feasible schedule, there always exists a normalized feasible schedule, and thus our restriction to these schedules is without loss of generality.

As every agent moves once to the left (to pick up the message – this move can be of zero length), and then once to the right, we have that in every normalized schedule every move of an agent to the right equals the advancement of the message done by this agent.

In the following we prove a structural lemma about normalized schedules, which will be a crucial ingredient in designing our algorithm. We will use the following notation. For an integer position p between s and t we say that an agent a (not necessarily from the schedule) is *p-crossing* if a lies at p or to the right of p, and at the same time the range of a allows the agent to walk left of p (to at least position $p-1$). By X_p we denote the set of all p-crossing agents. We denote by $A_p \subseteq X_p$ the agents of the schedule that are p-crossing, and that in the schedule never move left of p (i.e., they only move on the part of the line that is to the right of p). Thus, A_p are agents that could possibly help advancing the message in the part of the line to the left of p, but they do not (because they are used right of p).

Lemma 1. *Let p be a position (an integer) such that $s < p < t$. Then $|A_p| \leq 2 \cdot \log t$ (for $t \geq 2$), and $|A_p| \leq 2 \cdot \log(R_{max} + 1)$.*

Proof. Let $a_1, a_2, \ldots, a_\ell$ be the agents in A_p sorted in the order as they appear in delivering the message from s to t. Each agent a_i is responsible for advancing the message on a certain interval $I_i = [s_i, t_i]$ between p and t (recall that none of A_p moves left of p in the feasible schedule), where the order of the segments appearing on the line is identical to the order of the agents in which they move.

Recall that in a normalized schedule, agent a_i can stop before using all its range if it reaches the position of the next agent in the schedule. Let $t'_i \geq t_i$ be the point which a_i reaches if it uses all its range (to walk from s_i). It follows that

every two intervals $I'_i := [s_i, t'_i]$ and $I'_{i+2} := [s_{i+2}, t'_{i+2}]$ are disjoint (otherwise we can remove agent a_{i+1} from the schedule; a contradiction that the schedule is minimal). Thus, $s_{i+2} \geq t'_i + 1$. Furthermore, the position of a_{i+2} is (strictly) to the right of t'_i (as again we could remove agent a_{i+1} from the schedule).

Expressing t'_i in the form $p + \Delta_i$, we show that Δ_i grows exponentially with i and thus there can be at most (roughly) $\log t$ many agents in A_p before t'_i reaches t. Figure 2 illustrates the discussion of the proof. We start with t'_1: Clearly, agent a_1 can reach $p-1$ and thus, when reaching p, it can move at least to position $p+1$: $t'_1 \geq p+1$. We can continue with t'_3: Since $s_3 \geq t'_1 + 1$, we get $s_3 \geq p+2$; Furthermore, since agent a_3 is strictly to the right of t'_1, i.e., $a_3 \geq t'_1 + 1 \geq p+2$, and a_3 can reach $p-1$, a_3 has at $t'_1 + 1$ enough energy to move to the right to position $(t'_1+1)+((t'_1+1)-(p-1)) \geq (t'_1+1)+3 = t'_1+4$ (i.e., $t'_3 \geq p+5$). Similarly, $s_5 \geq t'_3 + 1$, agent a_5 lies to the right of t'_3, and agent a_5 has at t'_3+1 enough energy to walk to $(t'_3+1)+7 = t'_3+8$. In a similar spirit (i.e., by an easy induction), it follows that $t'_{2i-1} \geq t'_{2i-3} + 2^i$, $i = 2, 3, \ldots$. Thus, $t'_\ell \geq t'_1 + 2^2 + 2^3 + \ldots + 2^{\lceil \ell/2 \rceil} \geq (p+1) + (2^{\lceil \ell/2 \rceil + 1} - 4) = p + 2^{\lceil \ell/2 \rceil + 1} - 3$. By setting $t'_\ell \leq t$, we get $2^{\lceil \ell/2 \rceil + 1} \leq t + 3 - p \leq t + 2$, which implies $\ell/2 \leq \log(t+2) - 1$, which implies $\ell/2 \leq \log t$ (for $t \geq 2$), i.e., $\ell \leq 2 \cdot \log t$ (the first claim of the lemma).

At the same time, since a_ℓ is at least $t'_{\ell-2} + 1$, and the agent has energy to reach $p-1$, it follows that $R_\ell \geq p + 2^{\lceil \ell/2 \rceil} - 2 - (p-1) = 2^{\lceil \ell/2 \rceil} - 1$. Since the range of any agent is at most $R_{\max}$, we get $\ell \leq 2 \cdot \log(R_{\max} + 1)$. □

We now present our quasi-, pseudo-polynomial time algorithm. For simplicity of exposition, we will use the upper bound $|A_p| \leq 2 \log t$. We will further assume that $t \geq 2$ (to be able to apply Lemma 1): If $t = 1$, finding a solution is trivial, we just try every agent and see whether it can deliver the message on its own.

The main idea of our dynamic-programming based algorithm is to scan the line backwards from t to s and to gradually build a feasible schedule from the last agent delivering the message to t to the first agent picking the message at s and at every intermediate step p to remember the set $A_p \subseteq X_p$ of p-crossing agents that were used so far to the right of p, and thus are not available to be used to the left of p.

Formally, we define a boolean table $T[p, A_p]$ for every (integer) point p between s and t (including s and t) and every set $A_p \subseteq X_p$ of cardinality $|A_p| \leq 2 \cdot \log t$. We interpret the table as $T[p, A_p] = \texttt{true}$ if and only if there is a feasible schedule which advances the message from p to t, and among all p-crossing agents X_p it uses only the agents in A_p.

We fill the table as follows. We initialize $T[t, \emptyset] = \texttt{true}$. Then, for every $p = t-1, t-2, t-3 \ldots, s$ we enumerate all sets $A_p \subseteq X_p$ of cardinality $|A_p| \leq 2 \cdot \log t$ and set $T[p, A_p] = \texttt{true}$, if and only if

$$\exists p' > p,\ \exists A_{p'} \subseteq X_{p'},\ \exists \text{ agent } a_p \in A_p \setminus A_{p'} \text{ such that:}$$

$$a_p \text{ can bring the message from } p \text{ to } p', \tag{1}$$

$$A_p = \{a_p\} \cup (A_{p'} \cap X_p), \tag{2}$$

$$T[p', A_{p'}] = \texttt{true}. \tag{3}$$

After filling the table, the algorithm checks, whether for some set A_s there exists an entry $T[s, A_s] = \texttt{true}$, and if yes, it outputs a schedule of agents $a_{p_1}, a_{p_2}, a_{p_3}, \ldots, a_{p_\ell}$ that are, according to our dynamic program, recursively responsible for setting $T[p_i, A_{p_i}]$ to be `true` (this can be done by standard bookkeeping techniques); Otherwise, the algorithm decides that the agents cannot deliver the message from s to t.

Theorem 1. *The presented algorithm solves any instance of* DataDelivery *in time* $O\left(t^2 \cdot n^{1+4\log t}\right)$.

Proof. We will prove that there exists a solution to a given instance of DataDelivery if and only if the algorithm finds one.

If there is a solution to a given instance, i.e., a feasible schedule, then, by our observations, there also exists a normalized schedule $(a_{p_1}, a_{p_2}, \ldots, a_{p_\ell})$ of agents indexed with points on the line where they pick up the message, i.e., where agent a_{p_i} picks up the message at p_i and advances it to p_{i+1} (where we set $p_{\ell+1} := t$, and where, naturally, $p_1 = s$). Let S be the agents of this solution, i.e., $S = \{a_{p_1}, a_{p_2}, \ldots, a_{p_\ell}\}$. At any of these points p_i, $i = 1, 2, \ldots, \ell$, let $A_{p_i} \subseteq S \cap X_{p_i}$ be the set of agents from the solution S that are p_i-crossing. Furthermore, set $A_{\ell+1} = \emptyset$. Then, by Lemma 1, $|A_{p_i}| \leq 2 \cdot \log t$. Therefore, for every such set A_{p_i}, $i = 1, 2, \ldots, \ell+1$, there will be an entry $T[p_i, A_{p_i}]$ in our table. Furthermore, it follows that $A_{p_i} = \{a_{p_i}\} \cup (A_{p_{i+1}} \cap X_{p_i})$, $i = 1, \ldots, \ell$. Therefore, according to the rules of our dynamic programming (Eqs. (1), (2), (3)), all entries $T[p_i, A_{p_i}]$, $k = \ell, \ell-1, \ldots, 3, 2, 1$, will be set to `true` because of the previous entry $T[p_{i+1}, A_{p_{i+1}}]$. Thus, our algorithm finds a solution (e.g., the one just derived from the normalized feasible schedule).

Assume now that the algorithm finds a schedule $a_{p_1}, \ldots, a_{p_\ell}$ of agents, indexed by the points p_i at which the corresponding entry $T[p_i, A_{p_i}]$ was set to true. By the rules of filling the table, it follows that the agents can deliver the message from s to t. What remains is to argue that no two agents $a_{p_i}, a_{p_{i+\Delta}}$ from the returned schedule are the same agent. Assume, for the sake of contradiction, that this is the case, i.e., $a = a_{p_i} = a_{p_{i+\Delta}}$, and that Δ is the smallest such number. By Eq. (2), $a_{p_{i+\Delta}} \in A_{p_{i+\Delta}}$; At the same time, since $a \in X_{p_i}$ and because of Eq. (2), a appears in all sets $A_{p_{i+\Delta-1}}$, $A_{p_{i+\Delta-2}}$, $\ldots$, $A_{p_{i+1}}$, and especially in $A_{p_{i+1}}$. But this contradicts the fact that $a = a_{p_i} \in A_{p_i} \setminus A_{p_{i+1}}$.

The runtime of the algorithm follows from the size of the table T and the way we fill in the table: For every p, we enumerate at most $\sum_{i=1}^{2\log t} \binom{n}{i} \leq O(n^{2\log t})$ many sets A_p. To fill in an entry $T[p, A_p]$ we try all possible values p', $A_{p'}$ and a_p, and check whether conditions in Eqs. (1), (2), and (3) hold, which can be done in time linear in size of A_p, $A_{p'}$, and X_p (if we store the elements of the sets sorted according to their position on the line). This results in total running time of $O\left(t^2 \cdot n^{1+4\log t}\right)$ (with small constants hidden in the big-oh notation). □

Using the upper bound $|A_p| \leq 2 \cdot \log(R_{\max} + 1)$, we can bound the running time of the algorithm by $O\left(t^2 \cdot n^{1+4\log(R_{\max}+1)}\right)$.

3 NP-Completeness

We first create an auxiliary NP-hard problem Weighted-4-Partition, which we then reduce to DataDelivery. Along the way, we use the NP-hard problem 4-Partition-from-4-sets, which has been shown NP-complete in [6] as a step in proving the NP-hardness of 3-Partition.

4-Partition-from-4-sets

Input: Four sets of positive integers $A' = (a'_i)_{1\le i\le q}$, $B' = (b'_i)_{1\le i\le q}$, $C' = (c'_i)_{1\le i\le q}$, $D' = (d'_i)_{1\le i\le q}$, and an integer S'.
Question: Does there exist three permutations π_A, π_B, π_C of $[1, q]$ such that for every i, $a'_{\pi_A(i)} + b'_{\pi_B(i)} + c'_{\pi_C(i)} + d'_i = S'$?

Weighted-4-Partition

Input: A set of positive integers $E = (e_i)_{1\le i\le 4q}$ and an integer S such that for every partition of E into 4 sets A, B, C, D, each of size q, $\sum_{a\in A} a + \sum_{b\in B} 2b + \sum_{c\in C} 4c + \sum_{d\in D} 8d \le qS$.
Question: Does there exist a partition of E into q sets $E_1, E_2, \ldots, E_q$, each of size 4, such that for every $1 \le i \le q$, if $E_i = \{a, b, c, d\}$ with $a \le b \le c \le d$, $a + 2b + 4c + 8d = S$? We call such a partition a *weighted partition*.

Theorem 2. Weighted-4-Partition *is NP-hard.*

Proof. From an instance of 4-Partition-from-4-sets, we construct an instance of Weighted-4-Partition as follows. Note that we can assume that $\sum_{x'\in A'\cup B'\cup C'\cup D'} x' = qS'$, and that for every $x' \in A' \cup B' \cup C' \cup D'$, $x' \le S'$.

For each $i \in [1, q]$, let $a_i = 8a'_i$, $b_i = 4b'_i + 32S'$, $c_i = 2c'_i + 128S'$ and $d_i = d'_i + 512S'$. Let $A = \{a_i\}_{1\le i\le q}$, $B = \{b_i\}_{1\le i\le q}$, $C = \{c_i\}_{1\le i\le q}$ and $D = \{d_i\}_{1\le i\le q}$. Let $E = A \cup B \cup C \cup D$ and let $S = 4680S'$. Note that $\forall a \in A, b \in B, c \in C, d \in D, a < b < c < d$. Consequently, for any partition of E into 4 sets A^*, B^*, C^*, D^* of size q, $\sum_{a\in A^*} a + \sum_{b\in B^*} 2b + \sum_{c\in C^*} 4c + \sum_{d\in D^*} 8d \le \sum_{a\in A} a + \sum_{b\in B} 2b + \sum_{c\in C} 4c + \sum_{d\in D} 8d = \sum_{a'\in A'} 8a' + \sum_{b'\in B'}(8b' + 64S') + \sum_{c'\in C'}(8c' + 512S') + \sum_{d'\in D'}(8d' + 4096S') = 8\sum_{x'\in A'\cup B'\cup C'\cup D'} x' + 4672qS' = 4680qS' = qS$.

Suppose that we are given a solution π_A, π_B, π_C of 4-Partition-from-4-sets such that for every i, $a'_{\pi_A(i)} + b'_{\pi_B(i)} + c'_{\pi_C(i)} + d'_i = S'$. Then for every i, $a_{\pi_A(i)} + 2b_{\pi_B(i)} + 4c_{\pi_C(i)} + 8d_i = 8a'_{\pi_A(i)} + 8b'_{\pi_B(i)} + 64S' + 8c'_{\pi_B(i)} + 512S' + 8d'_i + 4096S' = 4672S' + 8(a'_{\pi_A(i)} + b'_{\pi_B(i)} + c'_{\pi_B(i)} + d'_i) = 4680S' = S$. Since $a_{\pi_A(i)} \le 8S' < 32S' \le b_{\pi_B(i)} \le 36S' < 128S' \le c_{\pi_C(i)} \le 130S' < 512S' \le d_i$, we have found a solution to Weighted-4-Partition, i.e., a weighted partition.

Conversely, suppose that there exists a partition $E_1, E_2, \ldots E_q$ that is a solution of the instance (E, S) of Weighted-4-Partition. For every $1 \le i \le q$, let $E_i = \{w_i, x_i, y_i, z_i\}$ with $w_i \le x_i \le y_i \le z_i$.

Suppose first that there exists $1 \le i \le q$ such that $E_i \cap D = \emptyset$. Then, $w_i \le x_i \le y_i \le z_i \le 130S'$, and thus $w_i + 2x_i + 4y_i + 8z_i \le 15 \cdot 130S' = 1930S' < 4680S'$, contradicting the fact that $E_1, E_2, \ldots E_q$ is a weighted partition of E. Consequently, for every $1 \le i \le q$, $z_i \in D$ and thus, $w_i, x_i, y_i \in A \cup B \cup C$.

Suppose now that there exists $1 \leq i \leq q$ such that $E_i \cap C = \emptyset$. Then, $w_i \leq x_i \leq y_i \leq 36S'$ and $z_i \leq 513S'$. Thus, $w_i + 2x_i + 4y_i + 8z_i \leq 7 \cdot 36S' + 8 \cdot 513S' = 4356S' < 4680S'$, a contradiction. Consequently, for every $1 \leq i \leq q$, $y_i \in C$ and thus, $w_i, x_i \in A \cup B$.

Suppose now that there exists $1 \leq i \leq q$ such that $E_i \cap B = \emptyset$. Then, $w_i \leq x_i \leq 8S'$, $y_i \leq 130S'$ and $z_i \leq 513S'$. Thus, $w_i + 2x_i + 4y_i + 8z_i \leq 3 \cdot 8S' + 4 \cdot 130S' + 8 \cdot 513S' = 4648S' < 4680S'$, a contradiction. Therefore, for every $1 \leq i \leq q$, $x_i \in B$ and thus, $w_i \in A$.

Consequently, each E_i contains one element of A, one element of B, one element of C and one element of D. Without loss of generality (by reordering the elements in D), assume that for every $1 \leq i \leq q$, $d_i \in E_i$. For every $1 \leq i \leq q$, let $\pi_A(i) = j$ iff $a_j \in E_i$, let $\pi_B(i) = j$ iff $b_j \in E_i$, and let $\pi_C(i) = j$ iff $c_j \in E_i$. Consequently, for every i, $a_{\pi_A(i)} + 2b_{\pi_B(i)} + 4c_{\pi_C(i)} + 8d_i = 4680S'$, i.e., $8a'_{\pi_A(i)} + 8b'_{\pi_B(i)} + 64S' + 8c'_{\pi_B(i)} + 512S' + 8d'_i + 4096S' = 4680S'$. Consequently, $4672S' + 8(a'_{\pi_A(i)} + b'_{\pi_B(i)} + c'_{\pi_B(i)} + d'_i) = 4680S'$ and thus $a'_{\pi_A(i)} + b'_{\pi_B(i)} + c'_{\pi_B(i)} + d'_i = S'$. Consequently, π_A, π_B, π_C is a solution of the instance (A', B', C', D', S') of 4-Partition-from-4-sets. □

Given an instance of Weighted-4-Partition (E, S) given by a set $E = \{e_1, e_2, \ldots, e_{4q}\}$ and by a target value S, we construct an instance of DataDelivery as follows. There will be two types of agents: $4q$ "big" agents corresponding to the elements of E, and $q + 1$ "small" agents.

Let $M = \max_i\{e_i\}$, and let $r = \max\{15M - S, \frac{32^q - 1}{31}S\}$. There is a small agent s_i starting at position $x_i = \frac{16^i - 1}{15}(S + 16r)$ for $0 \leq i \leq q$. The range of each small agent is r. Note that $x_0 = 0$ and that $x_{i+1} = 16(x_i + r) + S$. Note also that all the positions and ranges are integers. We set $s = 0$, and $t = x_q + r$. For each $1 \leq j \leq 4q$, there is a big agent b_j starting at position t with a range equal to $t + e_j$.

Note that every big agent b_j starts at t and its range enables it to reach any point between s and t. Moreover, if b_j collects the message at a point $l \in [s, t]$, the furthest point where it can deliver the message is $2l + e_j$.

We claim that there exists a weighted partition of (E, S) if and only if the set of agents $A = \{s_i\}_{0 \leq i \leq q} \cup \{b_j\}_{1 \leq j \leq 4q}$ can deliver the message from s to t. The first direction is straightforward, and shown in Lemma 2. The other direction is more complicated, and is shown after Lemma 2 in a series of claims.

Lemma 2. *If there exists a weighted partition $E_1, E_2, \ldots, E_q$ of E, then there is a feasible schedule for* DataDelivery *from s to t.*

Proof. Suppose that we are given 4 integers $a \leq b \leq c \leq d$ such that $a + 2b + 4c + 8d = S$. Consider four agents a', b', c', d' initially located on t with respective ranges $t + a, t + b, t + c, t + d$. We claim that if some message is on $x_i + r$ for $i < q$, then agents a', b', c', d' can move the message to x_{i+1} when they are activated in the following order: d', c', b', a'.

Since $x_i + r < x_q + r = t$, d' can move the message to $2(x_i + r) + d$. Since $2(x_i + r) + d < t$, c' can move the message to $4(x_i + r) + 2d + c$. Since $4(x_i + r) + 2d + c < t$,

b' can move the message to $8(x_i+r)+4d+2c+b$. Since $8(x_i+r)+4d+2c+b < t$, a' can move the message to $16(x_i+r)+8d+4c+2b+a = 16(x_i+r)+S = x_{i+1}$.

Recall that every small agent s_i, $0 \leq i \leq q$, can move the message from from x_i to $x_i + r$. Thus, we can use alternatively an agent s_i and the agents corresponding to E_{i+1} to move the message from $s = x_0 = 0$ to $t = x_q + r$. □

We now show the other direction. In the rest of the section, we assume that there exists a feasible schedule for the created instance of DataDelivery, and we show that there exists a weighted-partition of (E, S).

Up to rearranging the elements of E, assume that the big agents are activated in the order $b_1, b_2, \ldots, b_{4q}$. For every $1 \leq i \leq q$, let $B_i = \{b_{4i-3}, b_{4i-2}, b_{4i-1}, b_{4i}\}$ and let $\delta_i = 8e_{4i-3} + 4e_{4i-2} + 2e_{4i-1} + e_{4i} - S$. Note that for every i, $-S \leq \delta_i \leq 15M - S$.

Note that if we activate all agents from B_i consecutively (without activating a small agent in between) and if b_{4i-3} collects the message in $l \in [s, t]$, b_{4i-3} can deliver the message to $2l + e_{4i-3}$, b_{4i-2} can deliver the message to $4l + 2e_{4i-3} + e_{4i-2}$, b_{4i-1} can deliver the message to $8l + 4e_{4i-3} + 2e_{4i-2} + e_{4i-3}$, and b_{4i} can deliver the message to $16l + 8e_{4i-3} + 4e_{4i-2} + 2e_{4i-1} + e_{4i} = 16l + S + \delta_i$.

We denote by u_i the furthest point where, in the considered feasible schedule, b_{4i} can deliver the message. We denote by y_i the furthest point where s_i can deliver the message.

In the next two lemmas, we show that we can assume that for each $0 \leq i \leq q$, s_i is activated after b_{4i} and before b_{4i+1}.

Lemma 3. *For every $0 \leq i \leq q$, $y_i \leq x_i + r$. For every $1 \leq i \leq q$, s_i cannot be activated before b_{4i}, and $u_i \leq x_i + r$.*

Proof. The first assertion of the lemma is trivial since s_i starts in x_i and its range is r.

We prove the second assertion by induction on i. Let $i \geq 0$ and assume that $y_i \leq x_i + r$ and that $u_i \leq x_i + r$ if $i \geq 1$. Suppose that s_{i+1} is activated before b_{4i+4}.

Since $\max\{u_i, y_i\} \leq x_i + r$, and since $e_{4i+1}, e_{4i+2}, e_{4i+3} \leq M$, b_{4i+1} cannot deliver the message further than $2(x_i+r)+M$, b_{4i+2} cannot deliver the message further than $4(x_i + r) + 3M$, and b_{4i+3} cannot deliver the message further than $8(x_i + r) + 7M$. Since s_{i+1} cannot collect the message before $x_{i+1} - r = 16(x_i + r) + S - r = 16x_i + S + 15r$, it is enough to show that $7r + S > 7M$ in order to prove that s_{i+1} cannot be activated before b_{4i+4}. Since $r \geq 15M - S$ and $S \leq 15M$, $7r + S \geq 7 \cdot 15M - 6S \geq 15M > 7M$ and s_{i+1} cannot collect the message before b_{4i+4} has been activated.

Note that b_{4i+4} cannot deliver the message further than $u_{i+1} = 16(x_i + r) + 15M = x_{i+1} + 15M - S \leq x_{i+1} + r$. □

Lemma 4. *There exists a feasible schedule for* DataDelivery *from s to t such that for every $0 \leq i \leq q$, s_i is activated before b_{4i+1} and $y_i \geq x_i + r - \frac{2S}{31}(32^i - 1)$. For every $1 \leq i \leq q$, $u_i \geq x_i - \frac{S}{31}(32^i - 1) \geq x_i - r$.*

Proof. We prove the lemma by induction on i.

Note that since $x_0 = s$, $y_0 = x_0 + r$ (no matter when s_0 is activated). Suppose now that s_0 is not activated first. If when s_0 is activated, the message has already reached $x_0 + r$, then it means that there exists a feasible schedule for DataDelivery from s to t for $A' = A \setminus \{s_0\}$. Suppose now that when s_0 is activated, the message has not reached $x_0 + r$ and let i_0 be the maximal index i such that b_i has been activated before s_0. In this case, it means that there exists a feasible schedule for DataDelivery from $x_0 + r > s$ to t for $A' = A \setminus \{s_0, b_1, b_2, \ldots, b_{i_0}\}$. In both cases, it means that there exists a feasible schedule for DataDelivery from $x_0 + r > s$ to t for $A' = A \setminus \{s_0\}$ and thus there exists a feasible schedule for DataDelivery from s to t for A where s_0 is activated first.

Suppose now that s_i has been activated before b_{4i+1}, and that $y_i \geq x_i + r - \frac{2S}{31}(32^i - 1)$. Since s_{i+1} cannot be activated before b_{4i+4} (Lemma 3), we can assume that b_{4i+4} delivers the message to $u_{i+1} \geq 16y_i + S + \delta_{i+1} \geq 16(x_i + r) - \frac{32S}{31}(32^i - 1) = x_{i+1} - S - \frac{32S}{31}(32^i - 1) = x_{i+1} - \frac{S}{31}(32^{i+1} - 1)$. Since $r \geq \frac{S}{31}(32^{i+1} - 1)$, $u_{i+1} \geq x_{i+1} - r$.

Consequently, s_{i+1} can always be activated after b_{4i+4}. If s_{i+1} is activated before b_{4i+5}, either $u_{i+1} \geq x_{i+1}$ and $y_{i+1} = x_{i+1} + r \geq x_{i+1} + r - \frac{2S}{31}(32^{i+1} - 1)$, or $u_{i+1} < x_{i+1}$ and $y_{i+1} = 2u_{i+1} + r - x_{i+1} \geq x_{i+1} + r - \frac{2S}{31}(32^{i+1} - 1)$.

Suppose that we activate b_{4i+5} before s_{i+1}, then b_{4i+5} can deliver the message to a point $z \geq 2(x_{i+1} - r) + e_{4i+5} \geq x_{i+1} + (x_{i+1} - 2r) = x_{i+1} + (16x_i + 14r + S) > x_{i+1} + r$. That is, at this moment, s_{i+1} is useless. Consequently, there exists a feasible schedule for DataDelivery from z to t for $A' = A \setminus \{s_0, \ldots, s_i, s_{i+1}, b_1, \ldots, b_{4i+4}, b_{4i+5}\}$. This implies that there exists a schedule for DataDelivery from $2u_{i+1} + r - x_{i+1} \geq u_{i+1}$ to t for $A' \cup \{b_{4i+5}\}$, and thus, there exists a schedule DataDelivery from u_{i+1} to t for $A' \cup \{s_{i+1}, b_{4i+5}\}$ where s_{i+1} is activated first. □

From Lemmas 3 and 4, there exists a feasible schedule for DataDelivery from s to t where we activate alternatively a small agent and four big agents. Consequently, we can assume that for every $1 \leq i \leq q$, $u_i = 16y_{i-1} + S + \delta_i$ and that $y_i = 2u_i + r - x_i$ if $u_i < x_i$ and $y_i = x_i + r$ otherwise.

In the next lemma, we show that $\sum_{i=1}^{q} \delta_i \geq 0$ and that this inequality is strict if at least one small agent has to go back to collect the message.

Lemma 5. *For any two indices $i < j$ such that $y_i = x_i + r$, $u_l < x_l$ for every $i + 1 \leq l \leq j - 1$ and $u_j \geq x_j$, we have $\sum_{l=i+1}^{j} \delta_l \geq 0$. Moreover, this inequality is strict if $j > i + 1$.*

Proof. If $j = i + 1$, $u_{i+1} = 16(x_i + r) + S + \delta_{i+1} = x_{i+1} + \delta_{i+1}$ and consequently, $\delta_{i+1} \geq 0$. In the following, we assume that $j > i + 1$.

For every $i \leq l \leq j$, let $z_l = x_l + r - y_l$. Note that by Lemma 3, $z_l \geq 0$. Moreover, $z_i = z_j = 0$ and for every $i + 1 \leq l \leq j - 1$, $z_l > 0$.

For every integer $i \leq l \leq j - 1$, $u_{l+1} = 16y_l + S + \delta_{l+1}$. For $i \leq l \leq j - 2$, $u_{l+1} < x_{l+1}$ and thus, $y_{l+1} = 2u_{l+1} + r - x_{l+1} = 32y_l + 2S + 2\delta_{l+1} + r - x_{l+1}$.

Consequently, $z_{l+1} = x_{l+1} + r - y_{l+1} = 2x_{l+1} - (32y_l + 2\delta_{l+1} + 2S) = 32(x_l + r) + 2S - (32y_l + 2\delta_{l+1} + 2S) = 32(x_l + r - y_l) - 2\delta_{l+1} = 32z_l - 2\delta_{l+1}$. Thus, for every $i+1 \leq l \leq j-1$, $z_l = -2\sum_{t=i+1}^{l} 32^{l-t}\delta_t$.

Moreover, $u_j = 16y_{j-1} + S + \delta_j = 16(x_{j-1} + r) - 16z_{j-1} + S + \delta_j = x_j - 16z_{j-1} + \delta_j = x_j + \delta_j + 32\sum_{t=i+1}^{j-1} 32^{j-1-t}\delta_t = x_j + \sum_{t=i+1}^{j} 32^{j-t}\delta_t$.

Let $S_1 = u_j - x_j$ and $S_2 = \sum_{l=i+1}^{j-1} -\frac{z_l}{2}$, i.e., $S_1 = \sum_{t=i+1}^{j} 32^{j-t}\delta_t$ and $S_2 = \sum_{l=i+1}^{j-1}(\sum_{t=i+1}^{l} 32^{l-t}\delta_t)$. Since $u_j \geq x_j$, $S_1 \geq 0$, and since for every $i+1 \leq l \leq j-1$, $z_l > 0$, it follows that $S_2 < 0$. Consequently, $S_1 - 31S_2 > 0$. We claim that $\sum_{t=i+1}^{j} \delta_t = S_1 - 31S_2$. We get

$$\begin{aligned} S_1 - 31S_2 &= S_1 - 31\sum_{l=i+1}^{j-1}\sum_{t=i+1}^{l} 32^{l-t}\delta_t = S_1 - 31\sum_{t=i+1}^{j-1}\sum_{l=t}^{j-1} 32^{l-t}\delta_t \\ &= S_1 - 31\sum_{t=i+1}^{j-1} \frac{32^{j-t}-1}{31}\delta_t = \sum_{t=i+1}^{j} 32^{j-t}\delta_t - \sum_{t=i+1}^{j-1}(32^{j-t}-1)\delta_t \\ &= \sum_{t=i+1}^{j} \delta_t. \end{aligned}$$

Consequently, $\sum_{t=i+1}^{j} \delta_t = S_1 - 31S_2 > 0$. □

Proposition 1. *$(E_1, \ldots, E_q)$ is a weighted-partition of (E, S) where for each $1 \leq i \leq q$, $E_i = \{e_{4i-3}, e_{4i-2}, e_{4i-1}, e_{4i}\}$.*

Proof. Since for every partition of E into 4 sets A, B, C, D of size q, $\sum_{a\in A} a + \sum_{b\in B} 2b + \sum_{c\in C} 4c + \sum_{d\in D} 8d \leq qS$, $\sum_{i=1}^{q} \delta_i = \sum_{i=1}^{q}(8e_{4i-3} + 4e_{4i-2} + 2e_{4i-1} + e_{4i} - S) \leq 0$.

Since we have a feasible schedule for DataDelivery from $s = 0$ to $t = x_q + r$ where s_q is activated after b_{4q}, s_q delivers the message to $t = x_q + r$, and thus, $u_q \geq x_q$. By Lemma 5, $\sum_{i=1}^{q} \delta_i \geq 0$, and thus, $\sum_{i=1}^{q} \delta_i = 0$.

Moreover, if there exists $1 \leq i < q$ such that $u_i < x_i$, then $y_i < x_i + r$ and from Lemma 5, $\sum_{i=1}^{q} \delta_i > 0$, which is impossible. Consequently, for each $1 \leq i \leq q$, $u_i \geq x_i$ and $y_i = x_i + r$. Since $u_i = 16y_{i-1} + S + \delta_i = 16(x_{i-1} + r) + S + \delta_i = x_i + \delta_i$, it implies that for each i, $\delta_i \geq 0$.

Since $\sum_{i=1}^{q} \delta_i = 0$, we get that $\delta_i = 0$ for every $1 \leq i \leq q$, i.e., $8e_{4i-3} + 4e_{4i-2} + 2e_{4i-1} + e_{4i} = S$. Consequently $(E_1, E_2, \ldots, E_q)$ is a solution to the instance (E, S) of the Weighted-4-Partition problem. □

This ends the proof of the NP-hardness of DataDelivery. Note that one can check quickly whether a given permutation $\sigma = (a_1, \ldots, a_{n'})$ of a subset A' of the agents can solve an instance (A, s, t) of DataDelivery. Indeed, setting $t_0 = s$, agent a_{i+1} can reach t_i if and only if $a_{i+1} - R_{i+1} \leq t_i$. Moreover, if a_{i+1} can reach t_i, the furthest point agent a_{i+1} can reach with the information is $a_{i+1} + R_{a_{i+1}}$ if $t_i \geq a_{i+1}$ and $2t_i + R_{a_i} - a_i$ otherwise. Thus, one can iteratively compute the t_is until we find an index $t_i \geq t$, or until we find that $t_i < a_{i+1} - R_{i+1}$ or that

$t_{n'} < t$. This can be done by performing $O(n)$ arithmetical operations and the values we handle are smaller than $2t + R_{\max}$: this can be done in polynomial time and thus DataDelivery is in NP. Consequently, we get the following theorem.

Theorem 3. DataDelivery *is NP-complete.*

4 Conclusions and Open Problems

We have shown that DataDelivery on a line is NP-hard. This answers the open problem raised by Anaya et al. [3]. It actually is a surprising result, because everyone we talked to about the problem believed it to be polynomial. We accompanied the result with a quasi-, pseudo- polynomial time algorithm. It remains an open problem, whether a pseudo-polynomial time algorithm exists. It also is an interesting problem to provide good γ-resource augmented algorithms.

Acknowledgements. We are grateful for the valuable comments of the anonymous reviewers. Jérémie Chalopin acknowledges a partial support by ANR project MACARON (anr-13-js02-0002).

References

1. Albers, S., Henzinger, M.R.: Exploring unknown environments. SIAM Journal on Computing 29(4), 1164–1188 (2000)
2. Alpern, S., Gal, S.: The theory of search games and rendezvous, vol. 55. Kluwer Academic Pub (2002)
3. Anaya, J., Chalopin, J., Czyzowicz, J., Labourel, A., Pelc, A., Vaxès, Y.: Collecting information by power-aware mobile agents. In: Proc. 26th International Symposium on Distributed Computing (DISC). Lecture Notes in Computer Science, vol. 7611, pp. 46–60 (2012)
4. Blum, A., Raghavan, P., Schieber, B.: Navigating in unfamiliar geometric terrain. SIAM Journal on Computing 26(1), 110–137 (1997)
5. Chalopin, J., Das, S., Mihalák, M., Penna, P., Widmayer, P.: Data delivery by energy-constrained mobile agents. In: Proc. 9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS). pp. 111–122 (2013)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1979)
7. Heo, N., Varshney, P.K.: Energy-efficient deployment of intelligent mobile sensor networks. IEEE Transactions on Systems, Man, and CyberNetics (Part A) 35(1), 78–92 (2005)
8. Rajagopalan, R., Varshney, P.K.: Data-aggregation techniques in sensor networks: a survey. IEEE Communications Surveys & Tutorials 8(4), 48–63 (2006)