Data Delivery by Energy-Constrained Mobile Agents

Jérémie Chalopin[1], Shantanu Das[1], Matúš Mihalák[2], Paolo Penna[2], and Peter Widmayer[2]

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

Abstract. We consider mobile agents of limited energy, which have to collaboratively deliver data from specified sources of a network to a central repository. Every move consumes energy that is proportional to the travelled distance. Thus, every agent is limited in the total distance it can travel. We ask whether there is a schedule of agents' movements that accomplishes the delivery. We provide hardness results, as well as exact, approximation, and resource-augmented algorithms for several variants of the problem. Among others, we show that the decision problem is NP-hard already for a single source, and we present a 2-approximation algorithm for the problem of finding the minimum energy that can be assigned to each agent such that the agents can deliver the data.

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

1 Introduction

Recent progress in designing and producing small, simple, and cheap mobile micro-robots raised new algorithmic challenges in deploying these robots in various tasks. In this paper, we study the question of whether and how such simple robots can collaboratively transfer information from specified sources to a single repository. Due to their simplistic construction, the robots have only very limited capabilities, for example, little memory, small computational power, limited communication capabilities, noisy sensing, or limited battery power. In this paper, we focus on the last limitation aspect – the limited battery power. In particular, we study how such a limitation influences collective capabilities of the robots to accomplish the delivery task. We concentrate on this single aspect of the robots, and do not limit the other capabilities of the robots. In particular, we assume the robots to have enough memory to store the data, and we are also not interested in the amount of time it takes to finish. We study the delivery task on graphs; for this reason we adapt our terminology to the literature and refer to the robots as *agents*.

Model

We consider undirected, connected, edge-weighted graphs. The weight $w(e)$ of edge e represents the energy required to cross the edge. Therefore, we will sometimes refer to $w(e)$ as the *length* of the edge. By $d(u, v)$ we denote the distance between nodes u and v, i.e., the length of the shortest path from u to v (with respect to the edge weights).

We further consider k mobile agents that are initially placed on vertices of a given graph. Agent i can move along the edges of G. In total, agent i can move along a walk of length at most R_i. The agent can stop anywhere on an edge e. In such a case the travelled distance is proportional to $w(e)$ and to the position of the stop on e.

Furthermore, there are m distinct *sources* $S = \{s_1, \ldots, s_k\} \subset V$, and one *target* $t \in V \setminus S$. Each source contains data that needs to be delivered by the agents to target t. An agent i *collects data from source* s by simply visiting s on its walk. An agent i *collects data from agent* j by meeting agent j (at some location). An agent i visiting target t on its walk *delivers* (or *transfers*) all data that it has collected before.

We study the problem of deciding whether all data (from all sources) can be delivered to the target, i.e., whether there exists a schedule prescribing every agent how to move such that at the end all data is delivered. We call such a schedule *feasible*. In full generality, a schedule describes the movement of an agent in continuous time, assuming that all agents move at unit speed. We will see in a moment, however, that we may concentrate on schedules where at any time at most one agent moves. This then allows us to neglect the travel times and consider the movements of the agents in discrete time steps, where movements happen instantaneously.

We refer to the decision problem of finding a feasible schedule as DataDelivery. Given just the position of the agents in the network, we also study the related minimization problem of finding the smallest uniform power R for which the agents, when assigned the range R each, can deliver the data to t. We are interested in the computational complexity of the problem, and in approximation and resource-augmented algorithms. We say that an algorithm for the minimization version of DataDelivery is ρ-approximate, $\rho > 1$, if it runs in polynomial-time and always finds a feasible schedule for uniform range R such that $R \leq \rho \cdot R^*$, where R^* is the minimum uniform power for which a schedule exists. We say that an algorithm for the decision version of DataDelivery with agents' initial ranges R_i is a γ-resource augmented algorithm, $\gamma > 1$, if either the algorithm (correctly) answers that there is no feasible schedule, or it finds a feasible schedule for the modified (augmented) powers $R'_i := \gamma \cdot R_i$.

Related work

On a very high level, our problem can be seen as a special case of data aggregation in (wireless) sensor networks [10]. There, sensor nodes are deployed in an environment, each possessing some data that they need to route (transmit) over

an underlying communication network such that all data eventually arrives in a specific aggregation node. Obviously, the nature in which the data "flows" in the network makes the main difference of data aggregation in sensor networks to our problem.

There has been little previous work on data-aggregation-like problems by mobile agents. Anaya et al. [5] study the *convergecast* problem where a set of mobile agents, deployed in an edge-weighted graph, each possessing certain data and a uniform power R, need to move such that at the end at least one agent knows all data (and every agent travels a distance at most R). The main difference to our problem is that there are no sources and a target where the data need to be delivered. On contrary, in convergecast the "target agent" can be chosen freely to suit the given power constraints. Anaya et al. [5] study the convergecast problem both in the centralized and in the distributed setting. They show that the decision problem is strongly NP-complete, even if G is a tree, provide a linear-time algorithm for the case when G is a line, and a 2-approximation algorithm for the minimization version in general graphs. In the distributed setting, they provide a 2-approximate algorithm for trees and show that this is best possible (even if G is a line).

There is little research on general power-aware computation with mobile agents. A rare example is the study of self-deployment by Heo and Varshney [8]. Arguably, minimizing the total travelled distance (instead of the maximum traveled distance) by any single agent comes close to optimizing individual power-consumption. There is a rich research history accomplishing various tasks (such as pattern formation, exploration, or searching) by mobile agents where the prime optimization goal was the total travelled distance, see e.g. [6, 2, 3].

Power-aware computation is a relatively new research area. Most of the existing literature focuses on different computational models than mobile robots, e.g., on routing, tracking, and broadcasting in wireless networks [9, 4], or on scheduling [1, 7]. However, most of these works focus on minimizing the total energy consumption (whereas we focus on leveraging the consumed energy per computational entity).

Important observations and further variants

The nature of the problem allows us to make several crucial observations that limit the space in which we search for feasible strategies. We will argue about the single source case, but the very same observations can be made for the multi-source case as well.

First of all, it is easy to see that no two agents need to move at the same time. Assume that a given instance has a feasible solution and let us consider one. Let us consider the "flow" of the data from s to t in the solution, i.e., consider for every agent that collected the data the path that the agent made after the collection, and the union of all paths of the agents after they collected the information. Thus, this "flow" can be seen as the subgraph of G. It follows that there has to be an s-t path in the subgraph. Obviously, for completing the data delivery task, we can ignore all movements of the agents beyond this path.

Scanning this path from s to t and observing the identity of the agents that are currently active gives a sequence of agents (we do not need to choose more than one agent per position on the path). It is easy to see that the agents then can walk sequentially in this order, and thus we can only consider discrete time steps such that in each time step exactly one agents moves (to an arbitrary position).

It is now also easy to see that without loss of generality, no agent i appears more than once in this sequence: if yes, we can just ignore all agents that appeared in-between the two occurrences of agent i on the s-t path.

These considerations motivate the following natural variant of DataDelivery: Find a feasible schedule such that the data is moved from s to t along a *fixed* path (given as part of the input).

Our results

We first consider the single-source case in Section 2, and show that DataDelivery is NP-complete in this case, even for the case of uniform ranges R. We then provide a 3-resource augmented algorithm, and a 2-approximation algorithm for the problem. The combination of the ideas of these two algorithms provides a $\min\{3, (1+\Delta)\}$-resource augmented algorithm, where Δ is the largest ratio of the agent's ranges, i.e., $\Delta := \max_{i,j} \frac{R_i}{R_j}$. We also consider the case when the data needs to be moved along a fixed path P (given as part of the input), and show that also this problem is NP-complete, and that there exists $\gamma^* > 1$ such that there is no γ^*-augmented algorithm, unless $P = NP$. Finding a good approximation or resource-augmented algorithm for this version is left as an open problem. We also consider the special case when G is a line or a tree. If G is a line, we provide a polynomial-time algorithm for the case of uniform ranges. For the general (non-uniform) ranges, we leave the complexity of the problem open (and note that the $\min\{3, (1+\Delta)\}$-resource augmented algorithm applies). The case when G is a tree translates to the case of a line with general (non-uniform) ranges, and thus remains open as well.

We study the case of multiple sources in Section 3. For the constant number sources k and for general graphs, the natural adaptation of the results for single source carry over. For the general number of sources, the problem becomes NP-complete already for trees and for uniform ranges, by a trivial modification of the hardness result for convergecast by Anaya et al. [5].

2 Single source

In this section we study DataDelivery with single source node s. We first show the hardness result.

Theorem 1. *Deciding whether k agents can transfer the information from a given source s to a given target t is (strongly) NP-complete, even for unweighted graphs and for uniform ranges.*

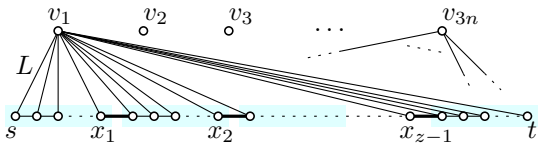

Fig. 1. Illustration of the reduction of 3-Partition to our problem. The horizontally aligned vertices from s to t form the dedicated path P^*. The shaded boxes along P^* are the buckets. Each line connecting v_1 with P^* stands for a path of length L. For simplicity, the lines connecting $v_2, \ldots, v_{3z}$ with P^* are omitted.

Proof. A solution to our problem is a set of walks, one for every agent, whose union forms a subgraph in which s and t are connected. Thus, our problem is obviously in NP – it is easy to check in polynomial time whether the given set of walks satisfy all required conditions.

To show the hardness, we will reduce the 3-Partition problem to our problem: given integers $a_1, a_2, \ldots, a_{3z}$, for some $z \in \mathbb{N}$, and an integer B such that $\sum_i a_i = z \times B$, the 3-Partition problem asks whether there is a partition $S_1, \ldots, S_z$ of the integers such that $\sum_{x \in S_i} x = B$. 3-Partition is NP-complete even if B is polynomially bounded in z, and if for every i, $B/4 < a_i < B/2$.

Given an instance of 3-Partition we construct an unweighted instance of our problem as follows. The graph contains a dedicated s-t path P^* of length $zB + (z-1)$. The first B edges on the path are called the *first bucket*. At the end of the first bucket, we place an auxiliary agent with range 1. This agent can thus help to transfer the message only along the adjacent edge on the s-t path. After this edge, the *second bucket* starts (containing again B edges), followed by a second auxiliary agent of range 1, and one edge, and the *third bucket* and so on. For every integer a_i we create a vertex v_i and connect it to every vertex of the s-t path by a path of length L. We place an agent to every vertex v_i and set its range to be $L + a_i$, where $L = 2B$. Figure 1 illustrates the reduction.

We now show that there is a solution to 3-Partition if and only if there is a solution in the just described instance of our problem. The "if" part is trivial: Given a solution of Partition, just use the three agents corresponding to the set S_i to move the data within bucket i. Use the auxiliary agents to advance the data on the edge between the buckets. We now argue about the "only if" part. We first show that the data needs to be transported along the dedicated path P^*. The only alternative is to transfer the data from s to vertex v_i, $i = 1, \ldots, z$, and from there to t. This path P_i has length $2L$. Obviously, agent i with range $R_i = L + a_i$ sitting at v_i cannot alone transfer the data along this path. Any other agent j can get to s (by travelling the distance L from v_j) and from there to distance at most $a_j < B/2$ from s on the path P_i. From there, no other agent but v_i can advance the data along the alternative P_i; the agent then can advance the data further to distance $a_j + ((L + a_i) - (L - a_j)) < B/2 + B$, i.e., to a position on P_i that is before vertex v_i. It is easy to see that no agent can further advance the data from there. Thus, the only way to transport the data is to use the dedicated s-t path P^*. Now, because the length of P^* is $zB + (z-1)$ and

because every agent i can advance the data on P^* by at most a_i steps, every solution to our problem needs to use all agents (including the auxiliary ones) in their "full power". Thus, such a solution uses exactly three agents in every bucket, bringing the data collectively from the beginning of the bucket to the end of the bucket. This then gives a solution to the 3-PARTITION.

We can easily modify the reduction for the case of uniform ranges R: just add a path of length $R - R_i$ to every vertex v_i and place the agent i at the end of the path. □

In the following we show that the hard part of the problem lies in knowing the order in which the agent move (and not in routing the agents through the graph). Namely, we show that given the order in which the agents move, we can solve in polynomial time whether there is a feasible schedule compatible with the given order.

Theorem 2. DATADELIVERY *with single source is solvable in polynomial time if restricted to a fixed order of the agents to move, and if agents can meet only at vertices.*

Proof. For each agent i we can compute a set of feasible "pick-pass" locations, that is, the set of all pairs (x, y) such that i can move to x (to pick up the information from another agent) and move to y (to pass the information to another agent),

$$C_i := \{(x, y) | \ \ d(i, x) + d(x, y) \leq R_i\}.$$

Given an ordered sequence of agents (expressing the order in which they need to advance the data), where each agent appears at most once, we can compute a feasible movement of the agents by looking at the following layered graph. Layer i_k contains the edges of C_{i_k} and a path from s to t in this graph corresponds to a feasible movement of the agents (every agent appears at most once and thus its movement is a single "pick-pass" edge which, by definition, is feasible for its range). Note that, since agents can only meet at the nodes of the graph, this layered graph can be computed in polynomial time. □

We now present a 3-augmented algorithm for DATADELIVERY with single source on general graphs and with general ranges. Our algorithm first checks whether it is (at all) possible that a feasible schedule exists. For this purpose, consider a *ball* $B(i)$ of radius R_i centered in the initial location of agent i, i.e., the set of all positions (vertices and positions on the edges) at distance at most R_i from i. If there is a feasible schedule, then there is one such that the data travels from s to t along a simple path, carried over by a sequence of ℓ agents $i_1, \ldots, i_\ell$ (and where no agent appears more than once in the sequence). Observe now that (1) s is in the ball of agent i_1 (i_1 is able to reach s to collect the data), (2) the balls of i_j and i_{k+1} intersect (agent i_j collects data from i_j), and (3) t is in the ball of the last agent i_ℓ (agent i_ℓ delivers the data to t). These properties imply the existence of an s-t path in the *connectivity graph*: the vertices are s, t and the agents, and there is an edge between i and j, if the balls $B(i)$ and

$B(j)$ intersect, and where we set $B(s) := \{s\}$ and $B(t) := \{t\}$. We can check the existence of an s-t path in the connectivity graph in polynomial time. If there is no such path, then there is no solution for DataDelivery. Otherwise, if there is such a path, we show that there is a feasible schedule for agents with new ranges $R'_i = 3 \cdot R_i$.

The feasible schedule for R'_i can be found in the following way. We first find an s-t path in the connectivity graph; recall that every agent appears at most once in this path. This path induces a natural order on the agents (that appear on the path), and let $i_1, \ldots, i_\ell$ be the order of these agents. For every two agents i_j and i_{j+1}, $j < \ell$, let x_j be an arbitrary vertex in $B(i) \cap B(i+1)$. Define further $x_0 := s$ and $x_\ell := t$. Then, every agent i_j moves as follows: it first goes to x_{j-1}, collects the data there, it goes back to initial position, and from there it goes to x_j. Obviously, with this schedule, the data gets delivered to t. Furthermore, every agent i_j does not travel more than $3 \cdot R_{i_j}$ (as every of its "three" moves are within its range R_i). We have thus proved the following.

Theorem 3. *There is a 3-resource augmented algorithm for* DataDelivery *with single source.*

The ideas of the 3-resource augmented algorithm can be adapted to give a 2-approximation algorithm for the optimization variant of DataDelivery with single source. Recall that in the optimization version, we are asked to find the minimum uniform range R such that there is a feasible schedule.

We will use the following observations. Consider an optimum solution, i.e., the smallest R^* and a corresponding schedule. Let i_1 be the first agent from the optimum solution to move, i.e., the agent that collects the data from s. Without loss of generality, we may assume that the optimum solution moves agent i_1 to s along a shortest path. This now induces a new instance of the problem: agent i_1 is now located in s, and has range $R' = R^* - d(i_1, s)$, while all other agents remain in their initial positions and with unchanged ranges R^*. By our construction, we know that this instance has a feasible schedule. This then implies that there is a path from i_1 (which sits on node s) to t in the connectivity graph of the modified instance.

The 2-approximation algorithm then works as follows. We first guess the first agent i_1 from the optimum solution that collects the data from s (i.e., technically, we try all possible candidate agents, perform the subsequent steps as explained below, and choose the solution giving the smallest range R among all the candidates). We move agent i_1 to s along a shortest path of length $d = d(i_1, s)$, and compute the smallest R^a such that there is a path from i_1 to t in the connectivity graph of the instance where every agent but i_1 has range R^a, and agent i_1 has range $R^a - d$. By the definition of R^a, we know that $R^* \geq R^a$. Let $i_1, i_2, \ldots, i_\ell, t$ be an i_1-t path in the connectivity graph of the considered instance. Thus, we know that for any $1 \leq j < \ell$, the respective balls intersect, and therefore $d(i_j, i_{j+1}) \leq 2 \cdot R^a$, and furthermore $d(i_\ell, t) \leq R^a$. Observe now that the schedule where agent i_j goes to agent i_{j+1}, $j < \ell$, and agent i_ℓ goes to t, is feasible if we add R^a to the range of every agent. This gives a feasible

schedule to the original setting where agent i_1 has not been moved to s, with uniform ranges $2 \cdot R^a$. We can thus return $2 \cdot R^a$ as the solution of the algorithm.

Because $R^* \geq R^a$, i.e., $2 \cdot R^* \geq 2 \cdot R^a$, we obtain that the algorithm is a 2-approximation.

Theorem 4. *There is a 2-approximation algorithm for* DataDelivery *with single source.*

Obviously, we can use the ideas of the 2-approximation algorithm for designing an equivalent 2-resource augmented algorithm for the case when the ranges are uniform, i.e., when $R_i = R_j$ for every i, j. The very same algorithm is then $(1+\Delta)$-resource augmented algorithm, where $\Delta = \max_{i,j} \frac{R_i}{R_j}$: It can happen that the algorithm decides for agent i to bring the data (from its initial position) to the initial position of agent j; For this, R_i needs to be increased by additive R_j to be able to do it, which gives the claimed ration $(1 + \Delta)$. Thus, we have the following.

Corollary 1. *There is a* $\min\{3, (1+\Delta)\}$*-resource augmented algorithm for* DataDelivery *with single source.*

We now consider a special case where the delivery of the data needs to happen along a fixed path in G. This is motivated by security reasons when we do not want the data to be delivered in dangerous areas of the environment. We now show that this problem is hard. We present an alternative proof for this case, since this gives us (additionally to the pure hardness result of Theorem 1) hardness for providing arbitrary good γ-resource augmented algorithms.

Theorem 5. *The variant of* DataDelivery *in which there is a single source and the data must travel along a fixed path of the graph is NP-hard.*

Proof. We reduce the problem from the restriction of 3-SAT in which every variable appears at most *four* times [11].

The idea of the reduction is as follows (see Fig. 2). Each clause consists of a "gadget" which has some common part with the fixed path. Intuitively speaking, if a clause is satisfied, then the data can travel from left to right along the portion of the path "covered" by that clause. The reduction will ensure that there is a satisfying assignment if and only if the data can travel from left to right through each of the clauses sub-paths.

More formally, for each clause $C_j = \{l_{j_1}, l_{j_2}, l_{j_3}\}$, we create a gadget consisting of a graph and an agent with range 5 as shown in Fig. 3 (upper part). For each variable x_i the corresponding gadget is the simple graph plus the agent of range R shown in Fig. 3 (lower part); We shall set $R < 3L$ so that this agent is forced to choose between "true or false". The three *edge literals* in the clause gadget are connected to some vertices of the corresponding variable gadget: The endpoints of an edge for literal x_i (resp., literal $\neg x_i$) are connected to the vertex in the variable gadget of x_i corresponding to $true$ (resp., $false$). The overall construction (see Fig. 2) consists in a concatenation of all clause gadgets, where

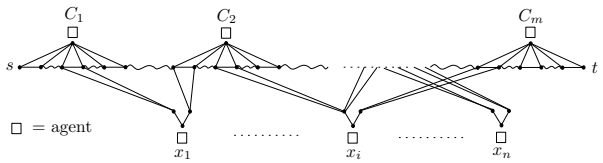

Fig. 2. Overview of the reduction from 3-SAT.

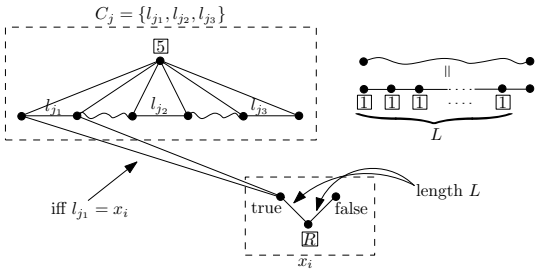

Fig. 3. The gadgets for clauses (upper part) and for variables (lower part) and how they are interconnected. An agent is shown as box with a number inside (its range). The two edges in the variable gadget have length L. Every other edge has unit length and the splines represent a chain of L edges with L additional agents of range 1 each.

any two consecutive gadgets are connected via a *spline path*, that is, a chain of L edges/agents like the one shown in Fig. 3.

We say that an agent *covers* and edge of the path if it traverses that edge from left to right. By setting $R = L + 8$ and $L = 8$ we have that

1. The agent for the clause cannot cover all three literal edges alone, but can cover any two of them. It is impossible for this agent to cover some edge of *another* clause gadget (because of the spline paths between clause gadgets).
2. The agent for a variable x_i can either (1) cover one edge in each of the clauses where x_i appears positive or (2) cover one edge in each of the clauses where x_i appears negated. It is impossible for this agent to cover an edge from a clause where x_i is positive *and* from another clause where x_i is negated.

Note that in the second item we use the fact that every variable appears in at most *four* clauses and, without loss of generality, it appears both positive and negated (so, there are at most three positive occurrences and at most three negated ones).

Claim. For every satisfying truth-assignment there exists a feasible movement of the agents such that the data travels along the path.

Proof (of Claim). If the assignment sets $x_i = true$ (resp., $x_i = false$) then, by Item 2, the variable agent can cover one literal edge in each of the clauses where x_i appears positive (resp., negated). Since the assignment satisfies all clauses, each clause has one literal edge covered by a variable agent. The remaining two literal edges can be covered by the clause agent of the clause (Item 1). The edges in the spline paths are covered by the corresponding agents. □

Claim. For every feasible movement of the agents such that the data travels along the path, there exists a satisfying truth-assignment.

Proof (of Claim). In every feasible movement all edges in the path must be covered by some agent. In particular, for each clause, there must be one of the three literal edges that is covered by a variable agent (Item 1). By Item 2 we can obtain a truth-assignment as follows: If the variable agent for x_i covers edge literals of clauses where x_i appears positive, then we set $x_i = true$; Otherwise we set $x_i = false$. By the previous argument, this assignment satisfies all clauses. □

The two claims above imply the NP-hardness. □

Note that the proof of hardness can be easily extended to the case of identical ranges. Moreover, with minor modifications, the reduction can be extended to prove that there is no γ-augmented algorithm for this variant of the problem, for some constant $\gamma > 1$.

Corollary 2. *There exists $\gamma > 1$ such that there is no γ-augmented algorithm for* DataDelivery *with single source and fixed delivery path, unless P=NP.*

3 Multiple sources

In this section we consider the version of DataDelivery in which the agents have to collect the data from *more than one source* to a common target location.

3.1 A 2-approximation for a constant number of sources with identical powers

The 2-approximation algorithm from Theorem 4 can be generalized to the case of a constant number of sources. Intuitively speaking, the algorithm guesses the set of "pick-up" agents that first reach the sources and, if an agent picks up data at more than one source, then it also guesses the *order* in which this is done (this is possible since there is only a constant number of sources).

More formally, in the optimal solution the piece of data at every source s_i travels along some path whose first agent is the *pick-up* agent of that source. Note that an agent can be the pick-up agent of several sources. Given the set

P of pick-up agents, each pick-up agent $p \in P$ is then matched to an ordered sequence of sources,

$$s_1^{(p)} \to s_2^{(p)} \to \cdots \to s_{\ell_p}^{(p)},$$

meaning that, in the optimal solution, agent p visits these sources in that particular order (possibly by visiting other locations in between). After being visiting the last source, p will move to some location to pass its data to some other agent i_p. Similarly to the case of a single source, we consider a new instance in which p has moved to its last source $s_{\ell_p}^{(p)}$ and its initial power R has been decreased by the minimum cost of visiting these sources in that particular order:

$$d(p, s_1^{(p)}) + d(s_1^{(p)}, s_2^{(p)}) + \cdots + d(s_{\ell_p-1}^{(p)}, s_{\ell_p}^{(p)}).$$

(Visiting other additional locations between two consecutive sources can only increase this cost.) The new instance is thus feasible and, in particular, the range of p when starting from the last source allows it to move inside the ball of i_p. Thus it can move directly to i_p if provided an extra power of R. Therefore, any path from s to t in the connectivity graph of the modified instance yields a 2-approximation: Each pick-up agent visits all of its sources in the specified order and then moves from the last source to the first agent in the path (by the previous argument this costs at most $2R$). Then the subsequent agents simply bring the collected data directly to the next agent in the path (again the cost is at most $2R$ since these agents have power R also in the modified instance).

The 2-approximation algorithm now suggests itself: Guess the set of pick-up agents and their ordered sequence of sources (there are only constantly many since the number of sources is constant), and then check if the connectivity graph of the modified instance contains a path from s to t (this is indeed the case when the guess in the first step is correct).

Theorem 6. *There is a 2-approximation algorithm for* DataDelivery *with a constant number of sources.*

3.2 On hardness and approximation of arbitrary number of sources

When there are many sources but each source initially contains an agent, then the problem is NP-hard even for identical powers. This follows from Theorem 4 in [5], which study a conceptually different problem, but the proof of hardness applies also to our problem. We note that one can easily extend the 2-approximation algorithm from Section 3.1 for this very special case.

4 Conclusions and Open Problems

In this work we have studied several variants of DataDelivery. This problem concerns how a set of energy-constrained agents can collectively move some data from a set of given locations to a common target. The problems turn out to be hard, and the hardness lies (essentially) in finding how and in which order

the agents should move to perform this task. On the positive side, we provide algorithms that find solutions which guarantee that no agent uses more than a constant factor the energy required by the optimum. It would be interesting to close the gap between our approximation and resource augmentation algorithms and hardness results. Are there any better algorithms? What is the complexity of DataDelivery on special graphs? The problem is open even for trees and for the following simple geometric version: the agents lie on a single line with source and target being the endpoints. We note that if all agents have uniform ranges, then the problem is solvable on the line (but remains open for trees): start with the closest agent (to the source); when advancing the data, never overtake a (previously) unused agent; recursively use the closest agent to advance it further. Finally, it would be interesting to obtain positive results for the variant in which the piece of data must travel along a fixed (given graph), or for the case of an arbitrary number of sources.

References

1. Susanne Albers and Hiroshi Fujiwara. Energy-efficient algorithms for flow time minimization. *ACM Trans. Algorithms*, 3(4), 2007.
2. Susanne Albers and Monika R. Henzinger. Exploring unknown environments. *SIAM Journal on Computing*, 29(4):1164–1188, 2000.
3. Steve Alpern and Shmuel Gal. *The theory of search games and rendezvous*, volume 55. Kluwer Academic Pub, 2002.
4. Christoph Ambühl. An optimal bound for the mst algorithm to compute energy efficient broadcast trees in wireless networks. In *Automata, Languages and Programming*, ICALP 2005, pages 1139–1150, Berlin, 2005. Springer.
5. Julian Anaya, Jérémie Chalopin, Jurek Czyzowicz, Arnaud Labourel, Andrzej Pelc, and Yann Vaxès. Collecting information by power-aware mobile agents. In *Proc. 26th International Symposium on Distributed Computing (DISC)*, volume 7611 of *Lecture Notes in Computer Science*, pages 46–60, 2012.
6. Avrim Blum, Prabhakar Raghavan, and Baruch Schieber. Navigating in unfamiliar geometric terrain. *SIAM Journal on Computing*, 26(1):110–137, 1997.
7. Ho-Leung Chan, Jeff Edmonds, Tak-Wah Lam, Lap-Kei Lee, Alberto Marchetti-Spaccamela, and Kirk Pruhs. Nonclairvoyant speed scaling for flow and energy. *Algorithmica*, 61(3):507–517, 2011.
8. N. Heo and P. K. Varshney. Energy-efficient deployment of intelligent mobile sensor networks. *IEEE Transactions on Systems, Man, and CyberNetics (Part A)*, 35(1):78–92, 2005.
9. Qun Li, Javed Aslam, and Daniela Rus. Online power-aware routing in wireless ad-hoc networks. In *Proc. 7th Annual International Conference on Mobile Computing and Networking*, MobiCom 2001, pages 97–107, New York, NY, USA, 2001. ACM.
10. Ramesh Rajagopalan and Pramod K. Varshney. Data-aggregation techniques in sensor networks: a survey. *IEEE Communications Surveys & Tutorials*, 8(4):48–63, 2006.
11. Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984.