Time-constrained Data Harvesting in WSNs: Theoretical Foundation and Algorithm Design

Lin Chen*, Wei Wang†, Hua Huang‡, Shan Lin‡
*Lab. Recherche Informatique (LRI-CNRS UMR 8623), Univ. Paris-Sud, 91405 Orsay, France. chen@lri.fr
†Dept. Inf. Sci. & Electron. Eng., Zhejiang Univ., Hangzhou, China 321000, wangw@zju.edu.cn
‡Dept. Electrical & Computer Eng., Stony Brook Univ., huanghua.yh@gmail.com, shan.x.lin@stonybrook.edu

***Abstract*—Data harvesting using mobile data ferries has recently emerged as a promising alternative to the traditional multi-hop transmission paradigm. The use of data ferries can significantly reduce energy consumption at sensor nodes and increase network lifetime. However, it usually incurs longer data delivery latency as the data ferry needs to travel through the network to collect data, during which some delay-sensitive data may become obsolete. Therefore, optimizing the trajectory of the data ferry with data delivery latency bound is important for this approach to be effective in practice. To address this problem, we formally define the *time-constrained data harvesting problem*, which seeks an optimal data harvesting path in a network to collect as much data as possible within a time duration. We first characterise the performance bound given by the optimal data harvesting algorithm and show that the optimal algorithm significantly outperforms the random algorithm, especially when network scales. Motivated by the theoretical analysis and proving the NP-completeness of the time-constrained data harvesting problem, we then devise polynomial-time approximation schemes (PTAS) and mathematically prove the output being a constant-factor approximation of the optimal solution.**

I. Introduction

In wireless sensor networks with limited energy supply, a critical concern is how the sensing data from individual sensors can be collected to the sink with minimum energy consumption. The traditional multi-hop forwarding paradigm suffers from high energy consumption of forwarding nodes, especially those near the sink. As an efficient alternative, data harvesting using mobile devices, also termed as data mules [1] or data ferries [2], has been proposed and implemented in several applications such as underwater environmental monitoring [3]. The core idea can be summarised as follows: a data ferry (e.g., robot, vehicle) travels across the sensor field and harvests data from sensor nodes while they are within each other's communication range, and later transfers the harvested data to the sink.

The use of data ferries in data harvesting can significantly reduce energy consumption at sensor nodes and thus increase network lifetime. However, as the data ferry can harvest data only when it travels close to the target node, it usually incurs longer data delivery latency, during which some delay-sensitive data may become obsolete. Therefore, optimizing the trajectory of the data ferry to limit or minimise data delivery latency is a primary concern for this approach to be effective in practice.

In this paper, we consider the trajectory optimisation problem in data collection applications for wireless sensor networks. This problem seeks an optimal data harvesting path to collect as much data as possible within a time duration. We call the problem *time-constrained data harvesting problem*. Specifically, our problem formulates the situation when delay-sensitive data are reported to the sink within certain amount of time before they become obsolete. We conducted theoretical analysis and designed efficient algorithm for the time-constrained data harvesting problem. We have proved that the time-constrained data harvesting problem is NP-complete. To address this problem, we have designed a polynomial-time approximation schemes (PTAS). That is, our devised data harvesting algorithm gives a constant-factor approximation of the optimal solution of the time-constrained data harvesting problem in polynomial time.

The contributions presented in this paper are naturally articulated as follows:

- We formulate the time-constrained data harvesting problem. We analytically characterise the performance bound of the optimal data harvesting algorithm. Our analysis demonstrates that in a network where nodes are randomly deployed with fixed density and the data ferry moves at constant speed, the quantity of harvested data does not scale with the number of nodes in the network under the random data harvesting algorithm, while this quantity scales logarithmically for the optimal algorithm design, indicating a significant performance gain when the network scales. Even though the trend is logarithmic, the gap can still be significant in large networks. In other words, a data harvesting algorithm not carefully chosen, such as randomly choosing a data harvesting path, can be very inefficient.
- Motivated by the theoretical analysis, we focus on the design of PTAS finding a constant-factor approximation of the optimal solution. We first give a formal proof on the NP-completeness of the time-constrained data harvesting problem by relating it to the well-known travel salesman problem (TSP) [4], for which there is no polynomial-time algorithm with an approximation factor better than $\frac{220}{219}$ [5].
- Given the complexity of the problem, we first study a specific scenario with non-overlapping neighborhoods, i.e., the network is sufficiently sparse such that the data ferry cannot harvest data from multiple sensors without changing its location. We then extend the analysis to the

978-1-4799-8381-0/15/$31.00 ©2015 IEEE

generic case with overlapping neighborhoods, i.e., the network is sufficiently dense such that the data ferry can harvesting data from multiple sensors without changing location. For both cases, we develop a methodology that relates the performance of topological paths to geometrical paths to design PTAS and mathematically prove the output being a constant-factor approximation of the optimal solution.

Despite our focus on the data harvesting problem, the generic problem formulation of our work makes the analysis methodology and obtained results broadly applicable to several engineering domains ranging from mobile charger scheduling, target monitoring to security patrolling, with a common generic objective of designing an optimal path such that a time-constraint utility function depending on the number of encountered targets is maximised.

The rest of the paper is organized as follows. We formulate the time-constraint data harvesting problem in Sec. III. In Sec. IV, we derive performance of the optimal data harvesting algorithm and the random algorithm, laying the theoretical foundation of the problem. In Sec. V, we first establish the NP-completeness of the time-constraint data harvesting problem, and then design PTAS for the problem. Sec. VI presents simulation analysis of the proposed PTAS. Sec. VII the paper.

II. Related Work

The problem we address and the methodology we employ are related to the following research fields.

A. Data ferry Assisted Data Harvesting

There is a large body of existing work on data ferry assisted data harvesting [6], [7], [8], [9], [10] (cf. [11] for a comprehensive survey). The problem we address is the optimisation of data harvesting trajectory of the data ferry, which is a hard problem in general, since we are constrained in both space (communication range between the data ferry and sensors) and time domain (limiting data harvesting latency). Existing solutions contour this difficulty by either using simple mobility and communication models [6], [7], [8], [9], [10] or assuming that the trajectory is already given [6].

The authors in [12], [2] address a similar problem of designing data harvesting path for data ferries to minimise the data harvesting latency under the constraint that all sensors are visited. The algorithms they propose are based on the well-known travel salesman problem (TSP) [4] and its variant TSP with neighbors (TSPN) [13]. However, our problem is different because TSP requires the path to pass all sensors while we seek the most profitable path to harvest maximum data given the time constraint. Our problem formulation complements the TSP formulation and is particularly pertinent when the network is large and it is impossible to the data ferry to traverse every node. Technically, as detailed in the main part of the paper, our problem requires an original study that cannot draw from existing results.

B. Mobile Charger Scheduling

Another similar problem is the mobile agent scheduling problem where a mobile charger needs to travel within the charging range of each sensor node to recharge them under the constraint of the battery life of sensor nodes, which is similar to the time constraint in our data harvesting problem. However, they rely on additional assumptions or simplifications to make the problem tractable. For example, the authors of [14] find out a near-optimum traveling path to recharge all sensor nodes using linear programming, assuming the traveling speed being infinite, and then remove this assumption and derive a bound of performance degradation. However, their algorithm implicitly assumes the travelling is fast enough. In our work, we remove these assumptions and analytically establish the performance properties of the proposed data harvesting algorithm.

III. Time-constrained Data Harvesting Problem

We consider a sensor network composed of n nodes, denoted by the set $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$, randomly distributed in an Euclidean square $[0, D]^2$. We are interested in the asymptotic scenario where both n and D are large with the node density $\lambda = \frac{n}{D^2}$ being a constant. In this case, the considered network converges in distribution to an infinite random geometric graph induced by a homogeneous Poisson point process with density λ [15]. Each node v_i has unit data message[1] to be harvested by a data ferry, denoted by s, moving at a constant speed. To harvest data generated by v_i, s should move into the communication range of v_i, which is modeled as a disk D_i centered at v_i with radius r. We call D_i the neighborhood of node v_i. By slightly abusing notations, we also use D_i to denote the border of the disk. For a path $P \in \mathcal{P}$ where $\mathcal{P}$ denotes the possible path set, we denote $d(P)$ the Euclidean length of P. We say that a path P covers a point M if there exists a point on P within distance r to M. In other words, if s moves along P, it can harvest the data generated by all the nodes that it covers. Denote $\Lambda(P)$ the number of nodes P covers.

We consider the data harvesting problem faced by s in which it seeks an optimal data harvesting path to harvest as much data as possible within a time duration T. The problem we address models the situation where delay-sensitive data should be reported to the sink within certain time in order to be further analysed. To make the notation concise, we let s move at unit speed and thus T is the maximum path length s can traverse before deposing the harvested data. The results obtained can be easily scaled to arbitrary speed by scaling the time duration T. Throughout our analysis, we are interested in the non-trivial case where $r \ll T$ and $2Tr \ll D^2$, i.e., the maximum path length is much larger than the communication range, while the area covered by a path of length T is much smaller than the network area. The time-constrained data harvesting problem is formalized as follows.

Problem 1 (Time-constrained Data Harvesting Problem)**.** *The time-constrained data harvesting problem is as follows:*

[1]The case where nodes generate multiple data messages can be tackled by devising the node generating m unit data messages to m virtual nodes at the same position, each generating unit data message.

maximize $\Lambda(P)$,
subject to $d(P) \leq T$.

That is, s seeks the optimal path $P^* \in \mathcal{P}$ of Euclidean length $d(P^*) \leq T$, along which it can harvest the maximum quantity of data. When there are more than one maximum, the optimal path P^* is the one with minimum Euclidean length.

We conclude this section by stating the following properties of P^* that will be useful in subsequent proofs and analysis.

Lemma 1 (Properties of P^*)**.** *Let $P^*(\tau)$ denote the optimal solution of Problem 1 with parameter $T = \tau$, the following properties hold:*

- *Monotonicity:* $\Lambda(P^*(\tau_1)) \leq \Lambda(P^*(\tau_2)),\ \forall \tau_1 \leq \tau_2$;
- *Scalability:* $\Lambda(P^*(\kappa\tau)) \geq \dfrac{\kappa\Lambda(P^*(\tau))}{1+\kappa},\ \forall \tau,\ \forall \kappa \in (0,1)$.

Proof: The monotonicity follows straightforwardly from the definition of P^*. We now prove the scalability of P^*. Let m denote the integer such that $m < \frac{1}{\kappa} \leq m+1$. Divide $P^*(\tau)$ into $m+1$ non-overlapping parts of length $\frac{\tau}{m+1}$ each, it follows from the pigeonhole principle that there exists at least one part, denoted as p, which covers at least $\frac{\Lambda(P^*(\tau))}{m+1}$ nodes. It follows from $m < \frac{1}{\kappa} \leq m+1$ and the monotonicity property that

$$\begin{aligned}\Lambda(P^*(\kappa\tau)) &\geq \Lambda\left(P^*\left(\frac{\tau}{m+1}\right)\right) \geq \Lambda(p)\\ &\geq \frac{\Lambda(P^*(\tau))}{m+1} \geq \frac{\kappa\Lambda(P^*(\tau))}{1+\kappa},\end{aligned}$$

which completes the proof.

The time-constrained data harvesting problem has a number of important variants. In some applications, we require that the data harvesting path to be a cycle or have predefined starting and end points; it is sometimes required to differentiate sensor nodes by giving weights to them (e.g., giving higher weights to sensors at key positions) and seek the path maximising the weighted sum of harvested data; furthermore, we may dispose multiple data ferries to for data harvesting. Many of these variants can be addressed using the framework established in this paper to design and optimise data harvesting path.

IV. Analysis of Optimal and Random Data Harvesting Algorithms

Aiming at laying theoretical foundation of the time-constrained data harvesting problem, this section studies the performance of the optimal data harvesting algorithm and a natural algorithm where the data harvesting path is randomly chosen.

A. Random Data Harvesting Algorithm

A simple data harvesting algorithm is to randomly choose a data harvesting path of length T. We call this algorithm random data harvesting algorithm, termed concisely as random algorithm. Our motivation of starting with the random algorithm is two-fold:

- It is a natural strategy and very easy to implement;
- It provides a reference for performance comparison for more sophisticated algorithms as well as the optimal one.

In our study, we are particularly interested in the following questions:

- What is the performance of the random algorithm?
- What is the performance of the optimal data harvesting algorithm?
- What is the performance degradation between the random and the optimal algorithms?

To answer the above questions, we consider a sensor network as a random geometric graph depicted in Sec. III where n nodes are placed uniformly at random in the area $[0, D]^2$. Note that this is a natural modeling choice as in sensor networks, especially large ones, we usually do not have control over the position of nodes. Theorem 1 establishes the performance of the random algorithm in terms of the average quantity of harvested data and its sharpness[2].

Theorem 1 (Performance of Random Data Harvesting Algorithm)**.** *Consider the random data harvesting algorithm where s randomly chooses a path P of length T, it holds that*

- $\mathbb{E}[\Lambda(P)] \leq O(\lambda r T)$;
- $\Pr\{\Lambda(P) \geq n^\epsilon \mathbb{E}[\Lambda(P)]\} \to 0$, *when* $n \to \infty,\ \forall \epsilon > 0$, *that is,* $\Pr\{\Lambda(P) = \Theta(n^\epsilon)\} \to 0$.

Proof: Recall the notation that a point M is covered by a path P if the minimum distance between any point on P and M is at most r, it is straightforward to see that the maximum Euclidean area covered by a path of length T is $A = 2rT + \pi r^2$. Recall that $r \ll T$, it then holds that

$$\mathbb{E}[\Lambda(P)] \leq \lambda A = \lambda(2rT + \pi r^2) \Longrightarrow \mathbb{E}[\Lambda(P)] \leq O(\lambda r T).$$

To prove the second part of the theorem, we use the following Markov's inequality [16].

Lemma 2 (Markov's Inequality)**.** *For any non-negative random variable X and any $a > 0$, it holds that*

$$\Pr\{X \geq a\} \leq \frac{\mathbb{E}[X]}{a}.$$

Regarding $\Lambda(P)$ as a random variable and letting $a = n^\epsilon \mathbb{E}[\Lambda(P)]$, by applying Markov's inequality, we have

$$\Pr\{\Lambda(P) \geq n^\epsilon \mathbb{E}[\Lambda(P)]\} \leq \frac{1}{n^\epsilon} \to 0,\ \text{when } n \to \infty,$$

which quantifies the sharpness of $\Lambda(P)$.

With Theorem 1, we are able to answer the first question posed in the beginning of this subsection:

- In average, the expected harvested data for the random algorithm does not scale with respect to either the population size n of the network or its geometrical size D;
- With high probability, we cannot expect better outcome than $\Theta(\lambda r T)$.

[2]Throughout the paper, we use the following asymptotic notations:

- $g_1(x) = O(g_2(x)) \Leftrightarrow \exists c > 0, \exists x_0, \forall x > x_0, |g_1(x)| \leq c|g_2(x)|$;
- $g_1(x) = \Omega(g_2(x)) \Leftrightarrow \exists c > 0, \exists x_0, \forall x > x_0, |g_2(x)| \leq c|g_1(x)|$;
- $g_1(x) = \Theta(g_2(x)) \Leftrightarrow \exists 0 < c_1 \leq c_2, \exists x_0, \forall x > x_0, c_1 g_2(x) \leq g_1(x) \leq c_2 g_2(x)$.

B. Optimal Data Harvesting Algorithm

Having derived the performance of the random algorithm, we proceed to investigate the performance of the optimal data harvesting algorithm, as stated in Theorem 2.

Theorem 2 (Performance of Optimum Algorithm)**.** *Let P^* denote the path of the optimal data harvesting algorithm, it holds that*

- $\mathbb{E}[\Lambda(P^*)] = \Theta\left(\frac{\log n}{\log\log n}\right)$;
- $\Pr\left\{\Lambda(P^*) = \Theta\left(\frac{\log n}{\log\log n}\right)\right\} \to 1$, *when* $n \to \infty$.

Proof: We prove the theorem in two steps.

Step 1: lower-bound of $\Lambda(P^*)$: we show that $\Pr\left\{\Lambda(P^*) = \Omega\left(\frac{\log n}{\log\log n}\right)\right\} \geq 1 - \frac{1}{n} \to 1$, when $n \to \infty$. Our proof uses the well-known results in the bins and balls problem stated in the following lemma for completeness.

Lemma 3 (Maximum Load in Bins and Balls Problem [17])**.** *When throwing m balls into $\Theta(m)$ bins, the max-loaded bin has $\Theta\left(\frac{\log m}{\log\log m}\right)$ balls with probability at least $1 - \frac{1}{m}$ when $m \to \infty$.*

We first construct "bins" in the following claim.

Claim 1. *A path of length T can cover all nodes in a square having sides of length $\sqrt{2rT}$.*

Proof: We prove the claim by constructing a zigzag path P covering all nodes in a square $[0, b]^2$:

- Start from $(0, r)$ and move straightly towards (b, r);
- Move straightly from (b, r) to $(b, 3r)$;
- Move straightly from $(b, 3r)$ to $(0, 3r)$;
- Repeat the above process until covering all nodes in $[0, b]^2$.

The total length of the zigzag path P can be computed after some elementary geometrical operations as

$$d(P) = b \cdot \left\lceil \frac{b}{2r} \right\rceil + 2r \cdot \left\lfloor \frac{b}{2r} \right\rfloor.$$

When $T \gg r$, it can be calculated that with $d(P) = T$, P can cover all nodes in a square with sides of length $b = \sqrt{2rT}$.

Armed with Claim 1, we now divide the network area $[0, D]^2$ into $\frac{D^2}{2rT}$ non-overlapping bins, each corresponding to a square of sides of length $\sqrt{2rT}$. Apply Lemma 3 by regarding nodes as balls, the path covering all nodes in the max-loaded bin covers at least $\Theta\left(\frac{\log n}{\log\log n}\right)$ nodes with probability $1 - \frac{1}{n}$. Hence for the optimal path P^*, it holds that

$$\Pr\left\{\Lambda(P^*) = \Omega\left(\frac{\log n}{\log\log n}\right)\right\} \geq 1 - \frac{1}{n} \to 1, \text{ when } n \to \infty,$$

which complete Step 1 of the proof.

Step 2: upper-bound of $\Lambda(P^*)$: we show that $\Pr\left\{\Lambda(P^*) = O\left(\frac{\log n}{\log\log n}\right)\right\} \to 1$, when $n \to \infty$. We first prove the following claim.

Claim 2. *Divide the area of $[0, D]^2$ into $B = \frac{D^2}{4r^2}$ non-overlapping small squares[3] of sides of length $2r$. It holds that a path of length T covers at most $\Theta\left(\frac{T}{2r}\right)$ small squares.*

Proof: The proof is straightforward by noticing that any curve of length $2r$ covers at most 4 small squares.

Let k denote the number of non-overlapping small squares covered by the optimal data harvesting path P^*, it follows from Claim 2 that $k = O\left(\frac{T}{2r}\right)$. Let b denote the number of nodes in a small square, it follows from Lemma 3 that

$$\Pr\left\{b = O\left(\frac{\log n}{\log\log n}\right)\right\} \geq \Pr\left\{b = \Theta\left(\frac{\log n}{\log\log n}\right)\right\} \geq 1 - \frac{1}{n}.$$

When $n \to \infty$, we have $k \ll n$; the necessary and sufficient condition of $\Lambda(P^*) = O\left(\frac{\log n}{\log\log n}\right)$ is that all small squares covered by P^* contains $O\left(\frac{\log n}{\log\log n}\right)$ nodes; hence we have

$$\Pr\left\{\Lambda(P^*) = O\left(\frac{\log n}{\log\log n}\right)\right\} = \left(\Pr\left\{b = O\left(\frac{\log n}{\log\log n}\right)\right\}\right)^k \geq \left(1 - \frac{1}{n}\right)^{\Theta\left(\frac{T}{2r}\right)} \to 1,$$

following $T \ll n$. This completes the second step of the proof.

Combining the two steps proves the sharpness result:

$$\Pr\left\{\Lambda(P^*) = \Theta\left(\frac{\log n}{\log\log n}\right)\right\} \to 1, \text{ when } n \to \infty.$$

To prove $\mathbb{E}[\Lambda(P^*)] = \Theta\left(\frac{\log n}{\log\log n}\right)$, we proceed as follows:

- Apply the result of Step 1 and notice the fact that $\Lambda(P^*) \geq 0$, we have $\mathbb{E}[\Lambda(P^*)] = \Omega\left(\frac{\log n}{\log\log n}\right)$;
- Apply the result of Step 2 and notice the fact that $\Lambda(P^*) \leq n$, we have

$$\mathbb{E}[\Lambda(P^*)] \leq \Theta\left(\frac{\log n}{\log\log n}\right)\left(1 - \frac{1}{n}\right)^{\Theta\left(\frac{T}{2r}\right)} + n\left[1 - \left(1 - \frac{1}{n}\right)^{\Theta\left(\frac{T}{2r}\right)}\right] \leq \Theta\left(\frac{\log n}{\log\log n}\right), \ n \to \infty.$$

Combining above analysis leads to $\mathbb{E}[\Lambda(P^*)] = \Theta\left(\frac{\log n}{\log\log n}\right)$.

C. Discussion

Comparing the performance of optimal and random data harvesting algorithms, we can observe that when the network scales, especially when $n \to \infty$, the optimal algorithm significantly outperforms the random one. Even though the trend is logarithmic not polynomial or exponential, the gap can still be significant in large networks. In other words, a data harvesting algorithm not carefully chosen, such as randomly choosing a harvesting path, can be very inefficient. The motivates our second part of work on the following fundamental question:

- *How to design efficient data harvesting algorithms that approaches the performance of the optimal algorithm (i.e., efficient algorithms solving Problem 1)?* Mathematically, by efficient algorithms we mean polynomial-time approximation schemes (PTAS).

[3]To make the analysis concise and clear, we treat $\frac{D}{2r}$ as integer. The analysis can be easily extended to cover the case where $\frac{D}{2r}$ is not integer.

Remark. *Theorem 2 establishes the performance of the optimal algorithm. However, it does not specify how the optimal path can be constructed given a network instance. Choosing the path as indicated in the first step in the proof of Theorem 2 only performs well in the average sense when a large number of instances are executed, but it cannot give the optimal path for a given network instance. In fact, as we will show in the next section by Theorem 3, the problem of constructing the optimal path as formulated in Problem 1 is NP-complete.*

V. Polynomial-time Approximation Scheme Design for Time-constrained Data Harvesting Problem

In this section, we design polynomial-time approximation data harvesting algorithms that approaches the performance of the optimal algorithm. To start, we first show that Problem 1 is NP-complete.

A. NP-completeness of Problem 1

Theorem 3 (NP-completeness of Time-constrained Data Harvesting Problem)**.** *Problem 1 is NP-complete.*

Proof: To prove the NP-completeness of Problem 1, we relate Problem 1 to the well-known travel salesman problem (TSP) for which there is no PTAS with an approximation factor better than $\frac{220}{219}$, unless $P = NP$ [5]. Specifically, we show that if Problem 1 can be solved in polynomial time, then we can construct a PTAS with an approximation factor better than $\frac{220}{219}$ for the TSP.

To that end, given any graph $G_t \triangleq (\mathcal{V}_t, \mathcal{E}_t)$ on which we need to solve the TSP, we instantiate Problem 1 by choosing r such that $r < \min_{e\in\mathcal{E}_t} d(e)$, for example, $r \to 0$. We consider the non-trivial case where $|\mathcal{V}_t| \geq 2$. In the following we show that if Problem 1 can be solved in polynomial time, we can develop an algorithm that solves TSP in polynomial time with an approximation factor better than $\frac{220}{219}$.

Before developing our algorithm, we prove the following property of Problem 1.

Claim 3. *Denote $P^*(\tau)$ the solution of Problem 1 on G_t with parameter $T = \tau$ and $r < \min_{e\in\mathcal{E}_t} d(e)$; let $t_{min} \triangleq \min_{e\in\mathcal{E}_t} d(e)$ and $t_{max} \triangleq \sum_{e\in\mathcal{E}_t} d(e)$, it holds that*

$$\Lambda(P^*(t_{min})) = 2 \text{ and } \Lambda(P^*(1.5t_{max})) = |\mathcal{V}_t|.$$

Proof: For the first part, it is easy to see that $\Lambda(P^*(t_{min})) = 2$ when $r < \min_{e\in\mathcal{E}_t} d(e)$. To show $\Lambda(P^*(t_{max})) \geq |\mathcal{V}_t|$, it suffices to notice that a spanning tree of G_t can be converted into a path using the famous 1.5-approximation algorithm for the TSP in [18]. Since the length of any spanning tree of G_t is upper bounded by t_{max}, we are sure to be able to find a path passing all nodes with the maximum length $1.5t_{max}$. In other words, $\Lambda(P^*(1.5t_{max})) = |V_t|$.

Now we construct the following algorithm. We iterate on a variable t from $t = t_{min} + \epsilon$ to $1.5t_{max}$ by increasing t by ϵ from one iteration to the next, where ϵ is a small constant chosen such that $\epsilon \leq \frac{t_{min}}{220}$. In each iteration, we solve Problem 1 with the constraint $d(P) \leq t$. It follows from Claim 3 and Lemma 1 (Monotonicity) that there exists $t_0 \in [t_{min}, 1.5t_{max}]$ such that $\Lambda(P_1) \leq |\mathcal{V}_t| - 1$ and $\Lambda(P_2) = |\mathcal{V}_t|$, where P_1 and P_2 are the solutions of Problem 1 with the constraints $d(P) \leq t_0 - \epsilon$ and $d(P) \leq t_0$, respectively.

It can be noted that

- the above algorithm runs in polynomial time;
- P_2 is a solution of the TSP on G with an approximation factor $\frac{t_0}{t_0-\epsilon}$ which is upper bounded by $\frac{220}{219}$ as $t_0 \geq t_{min} + \epsilon$ and $\epsilon \leq \frac{t_{min}}{220}$.

This result contradicts the fact that there is no PTAS solving the TSP with an approximation factor better than $\frac{220}{219}$ and proves the NP-completeness of Problem 1.

B. Polynomial-time Approximation Algorithm Design: Non-overlapping Neighborhood Case

Given the complexity of the time-constrained data harvesting problem, we first investigate a specific scenario where the neighborhoods of any two nodes are non-overlapped (i.e., $D_i \bigcap D_j = \emptyset, \forall v_i, v_j \in \mathcal{V}$) and develop a PTAS for Problem 1. We start by the following definition of topological path.

Definition 1 (Topological path)**.** *A path P_t is called a topological path in a graph if P_t is composed of uniquely the edges in the graph.*

Generically, we call a path *geometrical path*, denoted as P_g for presentation clarity, to emphasize that P_g is not necessarily a topological path as P_g may contain curves and may start and end at any point. Of course, a topological path is also a geometrical one, i.e., let $\mathcal{P}_g$ and $\mathcal{P}_t$ denote the sets of geometrical and topological paths, it holds that $\mathcal{P}_c \subset \mathcal{P}_g$.

The key element towards designing a PTAS for Problem 1 is to establish the relationship between geometrical and topological paths in terms of path length and number of covered nodes, the two metrics on which we are focused. This relationship is established in two steps:

- *Step 1*: We show that any geometrical path P_g can be approximated by a topological path P_t such that
$$d(P_t) = O(d(P_g)), \text{ and } \Lambda(P_t) = \Lambda(P_g).$$
- *Step 2*: We show that any topological path P_t can be approximated by a geometrical path P_g via a *geometrisation* procedure that we develop such that
$$d(P_g) = O(d(P_t)), \text{ and } \Lambda(P_g) \geq \Lambda(P_t).$$

We start by the first step to approximate a geometrical path P_g by a topological path P_t. By slightly abusing the notation, for a given path P, we reuse $\Lambda(P)$ to denote an ordered set of nodes covered by P[4]. Using this notation, a topological path P_t can be uniquely noted by $\Lambda(P_t)$. Given an ordered set of nodes $\mathcal{V}_g = \{v_1, v_2, \cdots, v_{|\mathcal{V}_g|}\}$, for any geometrical path P_g with $\Lambda(P_g) = \mathcal{V}_g$, we construct a topological path $P_t = \mathcal{V}_g$. It holds that $d(P_t) = O(d(P_g))$ and P_t covers all nodes in $\mathcal{V}_g$. Let the geometrical path P_g^* denote the geometrical path of minimum length among those covering $\mathcal{V}_g$, it holds that $d(P_t) = \Theta(d(P_g^*))$. This approximation result is mathematically formalised in Lemma 4, whose proof is detailed in the technical report [19].

[4]We denote the end node with the smaller ID as the source node.

Lemma 4. *Given an ordered set of nodes $\mathcal{V}_g$, $\forall P_g$, $\Lambda(P_g) = \mathcal{V}_g$, let $P_t = \mathcal{V}_g$, it holds that $d(P_t) = O(d(P_g))$. Particularly, let $P_g^* = \underset{\Lambda(P_g)=\mathcal{V}_g}{\operatorname{argmin}} d(P_g)$, it holds that $d(P_t) = \Theta(d(P_g^*))$.*

We then proceed to the second step to approximate a topological path P_t by a geometrical path P_g by introducing geometrisation, formally defined in the following.

Definition 2 (Geometrisation)**.** *Given a topological path P_t, the geometrisation procedure finds a geometrical path P_g that approximates P_t. By approximation we require that*

$$d(P_t) = \Theta(d(P_c)), \text{ and } \Lambda(P_t) \geq \Lambda(P_c).$$

Algo. 1 details the proposed geometrisation procedure, whose core part is further illustrated in Fig. 1. It is straightforward to see that $d(P_g) < d(P_t)$. One technical point worth commenting is how to find M_i on D_i such that $|\overline{M_{i-1}M_i}| + |\overline{M_i v_{i+1}}|$ is minimised (line 6). M_i can be efficiently found by using the following technique: consider the outside border of D_i as a mirror; let a light beam be emitted from M_{i-1} and then be reflected by D_i to reach v_{i+1}; it follows from the theory of optics that light always travels using the shortest path; hence M_i corresponds to the reflection point of the light beam on D_i and can be found geometrically by equalising the angle of incident and the angle of reflection.

Algorithm 1 Geometrisation

Input: Topological path P_t passing nodes in $\mathcal{V}_t$
Output: Geometrised path P_g

1: Denote the intersection point of v_1v_2 and D_1 by M_1;
2: **for** $i = 2$ to $|\mathcal{V}_t| - 1$ **do**
3: **if** $\overline{M_{i-1}v_{i+1}}$ covers D_i **then**
4: Denote the first intersection point between $\overline{M_{i-1}v_{i+1}}$ and D_i by M_i; // See Fig. 1 (left);
5: **else**
6: Find a point M_i on D_i such that $|\overline{M_{i-1}M_i}| + |\overline{M_iv_{i+1}}|$ is minimised; // See Fig. 1 (right);
7: **end if**
8: **end for**
9: Denote the intersection point of $\overline{M_{|\mathcal{V}_t|-1}v_{|\mathcal{V}_t|}}$ and $D_{|\mathcal{V}_t|}$ by $M_{|\mathcal{V}_t|}$;
10: Return $P_g = \{\overline{M_1M_2}, \cdots, \overline{M_{|\mathcal{V}_t|-1}M_{|\mathcal{V}_t|}}\}$;

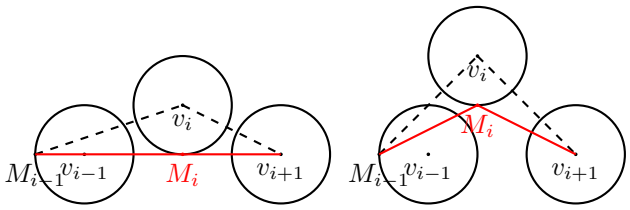

Fig. 1. Illustration of the core part of Algo. 1.

It is worth mentioning that the for loop in Algo. 1 can be repeated so as to further improve geometrisation effectiveness (i.e., decrease $d(P_g)$). To make this clearer, let $P_g^{j-1} = \{\overline{M_1^{j-1}M_2^{j-1}}, \cdots, \overline{M_{|\mathcal{V}_t|-1}^{j-1}M_{|\mathcal{V}_t|}^{j-1}}\}$ denote the output of Algo. 1 at iteration $j-1$, for iteration j, it suffices to set $P_t = P_g^{j-1}$ by letting $v_k = M_k^{j-1}$ $(2 \leq k \leq |\mathcal{V}_t| - 1)$ in the algorithm. We observe via simulation that that the improvement is not significant or even negligible when Algo. 1 is executed more than a handful of times.

After establishing the relationship between geometrical and topological paths, we are now ready to present the global PTAS for Problem 1, as detailed in Algo. 2.

Algorithm 2 PTAS solving Problem 1: non-overlapping neighborhood case

Input: Coordinates of nodes in $\mathcal{V}$
Output: Π^*: a constant factor approximation of P^*

1: Construct a complete graph G with node set $\mathcal{V}$; set the length of the edge between v_i and v_j to be $\overline{v_iv_j}$;
2: For each node pair (v_i, v_j), find the topological path $\Pi_t(i,j)$ passing the maximum number of nodes in $\mathcal{V}$ whose geometrised path $\Pi_g(i,j)$ satisfies $d(\Pi_g(i,j)) \leq T$ using Algo. 1 and the algorithm of max-prize path in [20] by setting the prize for each node to be 1;
3: Return $\Pi^* = \underset{\Pi_g(i,j),\forall v_i,v_j\in\mathcal{V}}{\operatorname{argmax}} \Lambda(\Pi_g(i,j))$;

The core idea of Algo. 2 is as follows: for each node pair, we find the topological path $\Pi_t(i,j)$ passing the maximum number of nodes in $\mathcal{V}$ whose geometrised path $\Pi_g(i,j)$ satisfies $d(\Pi_g(i,j)) \leq T$; we then return $\Pi^* = \underset{\Pi_g(i,j),\forall v_i,v_j\in\mathcal{V}}{\operatorname{argmax}} \Lambda(\Pi_g(i,j))$. The two building blocks in Algo. 2 is the geometrisation algorithm (Algo. 1) and the algorithm of max-prize path in [20]. Given a graph in which each node has a certain amount of prize, the max-prize algorithm finds in polynomial time a path collecting the maximum quantity of prize whose length is bounded by a constant, given as an input parameter. The following theorem formally establishes the performance of Algo. 2.

Theorem 4 (Performance of Algo. 2)**.** *Algo. 2 returns Π^* within polynomial time. It holds that $\Lambda(\Pi^*) = \Theta(\Lambda(P^*))$, where P^* denotes the optimal data harvesting path under time constraint T.*

Proof: The polynomial-time complexity of Algo. 2 follows from the polynomial-time complexity of Algo. 1 and the algorithm of max-prize path.

The second part of the theorem $\Lambda(\Pi^*) = \Theta(\Lambda(P^*))$ can be proved using Lemma 4 and Lemma 1. Specifically, it follows from Lemma 4 that for any $\tau \leq T$, there exists a topological path P_t such that

$$\Lambda(P_t) = \Lambda(P^*(\tau)), \ d(P_t) \leq c\tau,$$

where $c \geq 1$ is a constant factor. Now let $\tau = \frac{T}{c}$, apply Lemma 1 by setting $\kappa = \frac{1}{c}$, we have $d(P_t) \leq T$ and

$$\Lambda(P_t) = \Lambda(P^*(\tau)) \geq \frac{\Lambda(P^*(T))}{c+1}.$$

On the other hand, it follows from the geometrisation procedure and Algo. 2 that

$$\Lambda(\Pi^*) = \max_{\Pi_g(i,j),\forall v_i,v_j\in\mathcal{V}} \Lambda(\Pi_g(i,j)) \geq \Lambda(P_t) \geq \frac{\Lambda(P^*(T))}{c+1}.$$

The theorem is thus proved.

C. Polynomial-time Approximation Algorithm Design: Overlapping Neighborhood Case

In this subsection, we extend our efforts to study the generic case with overlapping neighborhoods.

We first construct a graph G' whose node set is $\mathcal{V}$ and there is an edge between v_i and v_j if $\overline{v_iv_j} \leq 2r$. We then construct a maximal independent set (MIS)[5] of G' using a coloring algorithm similar as presented in [21], [12], detailed in Algo. 3 for completeness.

Algorithm 3 MIS Construction of G'

Input: Graph G'
Output: MIS set $\mathcal{U}$
1: **Initialisation**: Set $\mathcal{U} = \emptyset$; Color all D_i ($v_i \in \mathcal{V}$) white;
2: **repeat**
3: Color a white disk D_i black and add v_i into $\mathcal{U}$;
4: Color every white disk D_j gray if v_j is v_i's neighbor;
5: **until** there is no white disk
6: Return $\mathcal{U}$;

We then define backbone topological paths, which can be regarded as topological paths using nodes in the MIS $\mathcal{U}$.

Definition 3 (Backbone Topological path)**.** *A path P_b is called a backbone topological path, or backbone path for short, in a graph if P_b is composed of uniquely the edges whose endpoints are in the MIS of the graph except the source and the destination nodes.*

As in the case of non-overlapping neighborhood, we call a path *geometrical path*, denoted as P_g, to emphasize that P_g is not necessarily a backbone path. Of course, a backbone path is also a topological path, and a geometrical one: i.e., let $\mathcal{P}_g$, $\mathcal{P}_t$ and $\mathcal{P}_b$ denote the sets of geometrical, topological and backbone paths, it holds that $\mathcal{P}_b \subset \mathcal{P}_t \subset \mathcal{P}_g$.

We apply the same analysis and design methodology in the non-overlapping neighborhood case and adapt it in the overlapping neighborhood case. A point M is said to be touched by path P if the minimum distance between any point of P and M is larger than r but smaller or equal to $2r$. The key element of designing a PTAS for Problem 1 with overlapping neighborhoods is to establish the relationship among geometrical, backbone, and geometrised backbone paths in terms of path length and number of touched and covered nodes. Specifically, we establish the relationship two steps:

- *Step 1*: We show that any geometrical path P_g can be approximated by a backbone path P_b such that $d(P_b) = O(d(P_g))$ and $\forall v_i$ covered by P_g, v_i is either covered or covered by P_b;
- *Step 2*: We show that any geometrical path P_g can be approximated by another geometrical path P'_g geometrised from a backbone path P_b via a *backbone geometrisation* procedure such that

$$d(P'_g) = O(d(P_g)), \text{ and } \Lambda(P'_g) \geq \Lambda(P_g).$$

[5]An independent set (IS) of an undirected graph is a subset $\mathcal{U}$ of nodes such that no two nodes in $\mathcal{U}$ are neighbors. An IS is maximal if no node can be added to $\mathcal{U}$ without violating IS. A maximal IS, or MIS, can be found in polynomial-time. Note that a related concept, a maximum IS (called MaxIS), is one IS of maximum cardinality. Finding MaxIS, however, is NP-complete.

We start with the first step by showing the following lemma. The proof uses similar reasoning technique as the proof of Lemma 4 and is detailed in the technical report [19].

Lemma 5. *Given any geometrical path P_g, there exists a backbone path P_b such that $d(P_b) = O(d(P_g))$ and $\forall v_i$ covered by P_g, v_i is either covered or touched by P_b. Particularly, let $P_g^* = \underset{\Lambda(P_g)=\mathcal{V}_g}{\operatorname{argmin}} d(P_g)$, it holds that $d(P_b) = \Theta(d(P_g^*))$.*

We then proceed to approximate a backbone path P_b by a geometrical path P_g by introducing *backbone geometrisation*, formally defined in the following.

Definition 4 (Backbone Geometrisation)**.** *Given a backbone path P_b, the backbone geometrisation procedure finds a geometrical path P_g that approximates P_b. By approximation we require that $d(P_b) = \Theta(d(P_g))$, and $\Lambda(P_b) \geq \Lambda(P_g)$.*

In [12], the authors develop a polynomial-time backbone geometrisation algorithm, which will be used in our design.

The following lemma approximates a geometrical path by another geometrical path geometrised from a backbone path.

Lemma 6. *Given any geometrical path P_g, there exists a path P'_g geometrised from a backbone path P_b such that*

$$d(P'_g) = O(d(P_g)), \text{ and } \Lambda(P'_g) \geq \Lambda(P_g).$$

Proof: The lemma follows straightforwardly from Lemma 5 and the backbone geometrisation algorithm.

After establishing the relationship among geometrical, backbone and geometrised backbone paths, we now present the design of the global PTAS for Problem 1 for the overlapping neighborhood case, as detailed in Algo. 4.

Algorithm 4 PTAS solving Problem 1: overlapping neighborhood case

Input: Coordinates of nodes in $\mathcal{V}$
Output: Π^*: a constant factor approximation of P^*
1: Construct a graph G' whose node set is $\mathcal{V}$ and there is an edge between v_i and v_j if $\overline{v_iv_j} \leq 2r$;
2: Run Algo. 3 on G' to construct an MIS $\mathcal{U}$;
3: Construct a complete graph G with node set $\mathcal{V}$; set the length of the edge between v_i and v_j to be $\overline{v_iv_j}$;
4: For each node pair (v_i, v_j), with the MIS $\mathcal{U}$ constructed in 2, find the backbone path $\Pi_b(i,j)$ passing the maximum number of nodes in $\mathcal{V}$ whose geometrised path $\Pi_g(i,j)$ satisfies $d(\Pi_g(i,j)) \leq T$ using the algorithm in [12] and the algorithm of max-prize path in [20] by setting the prize for each node i to be the number of nodes covered or touched by D_i;
5: Return $\Pi^* = \underset{\Pi_g(i,j),\forall v_i,v_j\in\mathcal{V}}{\operatorname{argmax}} \Lambda(\Pi_g(i,j))$;

The core idea of Algo. 4 is as follows: for each node pair (v_i, v_j), we find the bcckbone path $\Pi_b(i,j)$ passing the maximum number of nodes in $\mathcal{V}$ whose geometrised path $\Pi_g(i,j)$ satisfies $d(\Pi_g(i,j)) \leq T$; we then return $\Pi^* = \underset{\Pi_g(i,j),\forall v_i,v_j\in\mathcal{V}}{\operatorname{argmax}} \Lambda(\Pi_g(i,j))$. The two building blocks in Algo. 2 is the backbone geometrisation algorithm [12] and the

algorithm of max-prize path [20]. When running the algorithm of max-prize path, we set the prize of each node v_i to be the number of nodes covered or touched by D_i, which allows us to achieve constant-factor approximation (as detailed in the proof of Theorem 5). The following theorem establishes the performance of Algo. 4.

Theorem 5 (Performance of Algo. 4)**.** *Algo. 4 returns* Π^* *within polynomial time. It holds that* $\Lambda(\Pi^*) = \Theta(\Lambda(P^*))$*, where* P^* *denotes the optimal data harvesting path.*

Proof: The polynomial-time complexity of Algo. 2 following from the polynomial-time complexity of the geometrisation procedure [12] and the algorithm of max-prize path. We now prove second part of the theorem $\Lambda(\Pi^*) = \Theta(\Lambda(P^*))$.

Given a path P_g geometrised from a backbone path P_b, we denote the total collected prize along P_b by $Q(P_b)$ and set $Q(P_g) = Q(P_b)$. It can be noted that $\Lambda(\Pi^*) \leq Q(\Pi^*)$. It then follows from Algo. 4 and Lemma 6 that $Q(\Pi^*) = \Omega(\Lambda(P^*))$.

In the calculation of the max-prize in Algo. 4 (Step 4), a node may be counted multiple times in the final prize of the path. This is because a node can be covered by at most one node from the MIS $\mathcal{U}$ but can be touched by multiple nodes from $\mathcal{U}$. We next upper-bound the number of times a node is counted by 5. To prove this, we note that a node is counted more then once in the prize of a path if and only if it is not covered by any node in $\mathcal{U}$ and it is touched by multiple nodes in $\mathcal{U}$. Since any two node in $\mathcal{U}$ do not have overlapping neighborhoods, it is geometrically easy to see that any node not covered by any node in $\mathcal{U}$ can be touched by at most 5 nodes in $\mathcal{U}$. This result leads to $5\Lambda(\Pi^*) \geq Q(\Pi^*)$.

We have already proved that $Q(\Pi^*) = \Omega(\Lambda(P^*))$. It then holds that $\Lambda(\Pi^*) = \Omega(\Lambda(P^*))$. On the other hand, by the definition of P^*, we have $\Lambda(\Pi^*) \leq \Lambda(P^*)$. It then holds that $\Lambda(\Pi^*) = \Theta(\Lambda(P^*))$, which completes the proof.

VI. Numerical Analysis

In this section, we conduct numerical analysis to evaluate the performance of the our constant-factor approximation algorithm of the time-constrained data harvesting problem. To our knowledge, our algorithm is the only one addressing the trajectory optimisation in the time-constraint data harvesting problem, so we evaluate the performance of our algorithm with respect to the random algorithm where the data harvesting path is randomly chosen.

Specifically, we set up a simulation area of an Euclidean quare $[0, 1000]^2$ and randomly deploy a number of n nodes in the square, where n varies from 200 to 1000. The time constraint T is set to 100. We vary the communication range r to study various representative scenarios. By varying n and r, we can simulate both a sparsely deployed network where the neighborhoods of nodes are largely non-overlapping (small n and r) and a densely deployed network where the neighborhoods of nodes are largely overlapping (large n and r). For each set of chosen parameters, we run a number of independent simulations where the nodes' positions are randomly chosen and the required number of simulation runs is calculated using "independent replications" [22]. Throughout our simulations, we trace the following metric to evaluate the performance of our algorithm:

$$\Upsilon = \frac{\text{Quantity of data harvested by our algorithm}}{\text{Quantity of data harvested by random algorithm}}$$

The value of Υ characterises the performance gain of our algorithm over the random one. We are particularly interested in tracing the following cases:

- *Worst-case performance gain*: Under given parameters n, r, we study the worst-case performance gain among the simulation runs, i.e., the minimum value of Υ, denoted as Υ_{min}. This result gives the lower-bound of the performance gain our algorithm can achieve over the random one;
- *Best-case performance gain*: Under given parameters n, r, we study the best-case performance gain among the simulation runs, i.e., the maximum value of Υ, denoted as Υ_{max}. This result gives the upper-bound of the performance gain our algorithm can achieve;
- *Average performance gain*: Under given parameters n, r, we study the average performance gain among the simulation runs, i.e., the average value of Υ, denoted as Υ_{avg}. This result gives the average of the performance gain of our algorithm.

The simulation results are illustrated in Fig. 2 and Fig. 3. In Fig. 2, we fix $n = 200$ and trace Υ as a function of r by varying r from 2 to 10. In Fig. 2, we fix $r = 6$ and trace Υ as a function of n by varying n from 200 to 1000. Based on the simulation results, we make the following observations:

- Compared to the random algorithm, our algorithm achieves significant performance gain. Particularly, our algorithm can secure a performance gain of nearly 5 times that of the random algorithm in the simulated scenarios. In the extreme case, it performances 50 times better than the random algorithm. The results also demonstrate our theoretical finding in Sec. IV that a data harvesting algorithm not carefully chosen, such as randomly choosing a data harvesting path, may lead to significant performance loss.
- When the system scales, the performance gap between our algorithm and the random one increases, which again is in accordance of our theoretical analysis Sec. IV. When the communication range r increases, the the performance gap also increases. This can be explained by the fact that our algorithm carefully chooses the data harvesting path so as to cover as many nodes as possible given the time constraint. In contrast, the random algorithm cannot fully take the advantage brought by larger r with a randomly chosen path.

VII. Conclusion and Perspectives

In this paper, we have studied the problem of time-constrained data harvesting problem in which a data ferry seeks an optimal data collection path to collect as much data as possible within a time duration. This problem models the situation where time-sensitive data should be reported to the sink within certain time before they become obsolete. We have first characterised the performance bound given

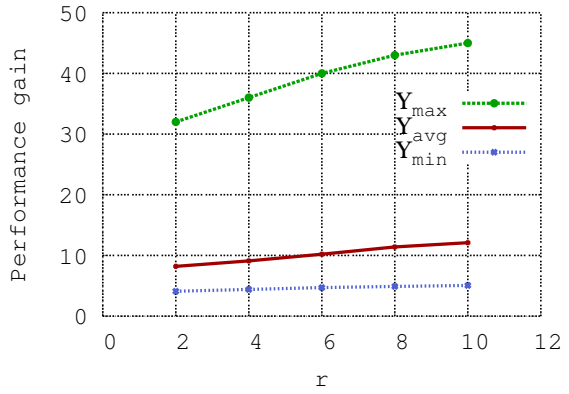

Fig. 2. Maximum, average, and minimum performance gain of our algorithm over the random algorithm as functions of r ($n = 200$).

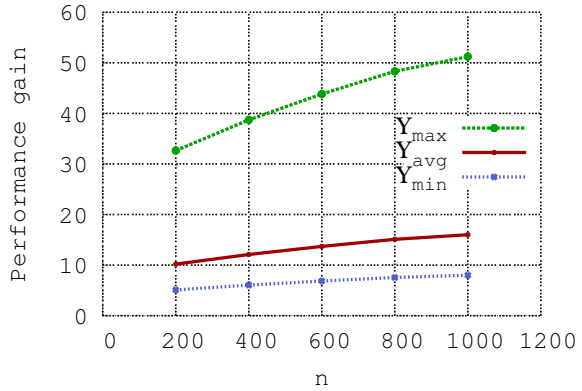

Fig. 3. Maximum, average, and minimum performance gain of our algorithm over the random algorithm as functions of n ($r = 6$).

by the optimal data harvesting algorithm and shown that the optimal algorithm significantly outperforms the random algorithm, especially when the network scales. Motivated by the theoretical analysis and proving the NP-completeness of the time-constrained data harvesting problem, we have then devised a PTAS of the problem and mathematically proved its output being a constant-factor approximation of the optimal solution.

As a small step towards characterising efficient data harvesting algorithms, our work can stimulate further investigations in this field. The first interesting research direction is to use the methodology in the paper to study more sophisticated variants of the data harvesting problem, e.g., the case of multiple data ferries with heterogeneous moving speed. The second consists of investigating the data harvesting problem where the data ferry does not have full knowledge of the network topology and should make its decision based on only local information and interactions.

VIII. Acknowledgement

The work of L. Chen is supported by the ANR grant Green-Dyspan (ANR-12-IS03). The work of W. Wang is supported by the NSFC grant 61261130585. The work of L. Shan and H. Huang is supported by the grants NSF CNS-1239108, CNS-1218718 and IIS-1231680.

References

[1] R. Shah, S. J. S. Roy, and W. Brunette, "Data mules: Modeling a three-tier architecture for sparse sensor networks," in *Proc. IEEE SNPA Workshop*, May 2003.
[2] L. Xue, D. Kim, Y. Zhu, D. Li, W. Wang, and A. O. Tokuta, "Multiple heterogeneous data ferry trajectory planning in wireless sensor networks," in *Proc. INFOCOM*, 2014.
[3] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke, "Data collection, storage, and retrieval with an underwater sensor network," in *Proc. ACM SenSys*, 2005.
[4] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. Harvard University Press, 2006.
[5] C. H. Papadimitriou and S. Vempala, "On the approximability of the traveling salesman problem," *Combinatorica*, vol. 1, no. 26, pp. 101–120, 2006.
[6] R. Sugihara and R. K. Gupta, "Speed control and scheduling of data mules in sensor networks," *ACM Trans. Sen. Netw. (TOSN)*, vol. 7, Aug. 2010.
[7] M. Ma and Y. Yang, "Sencar: An energy-efficient data gathering mechanism for large-scale multihop sensor networks," *IEEE Trans. Parallel Distrib. Syst. (TPDS)*, vol. 18, Oct. 2007.
[8] B. Yuan, M. Orlowska, and S. Sadiq, "On the optimal robot routing problem in wireless sensor networks," *IEEE Trans. on Knowl. and Data Eng. (TKDE)*, vol. 19, Sept. 2007.
[9] D. Ciullo, G. Celik, and E. Modiano, "Minimizing transmission energy in sensor networks via trajectory control," in *Proc. WiOpt*, 2010.
[10] G. Xing, T. Wang, W. Jia, and M. Li, "Rendezvous design algorithms for wireless sensor networks with a mobile base station," in *Proc. MobiHoc*, 2008.
[11] E. Ekici, Y. Gu, and D. Bozdag, "Mobility-based communication in wireless sensor networks," *IEEE Communications Magazine*, vol. 44, pp. 56–62, July 2006.
[12] D. Kim, B. Abay, R. N. Uma, W. Wu, W. Wang, and A. Tokuta, "Minimizing data collection latency in wireless sensor network with multiple mobile elements," in *Proc. IEEE INFOCOM*, 2012.
[13] A. Dumitrescu and J. Mitchell, "Approximation algorithms for tsp with neighborhoods in the plane," in *Proc. SODA*, 2001.
[14] L. Xie, Y. Shi, Y. T. Hou, W. Lou, and H. D. Sherali, "On traveling path and related problems for a mobile station in a rechargeable sensor network," in *Proc. MobiHoc*, 2013.
[15] M. Penrose, *Random Geometric Graphs*. Oxford University Press, 2003.
[16] O. Goldreich, *Computational Complexity*. Cambridge University Press, 2008.
[17] M. Mitzenmacher, *The power of two choices in randomized load balancing*. PhD thesis, Berkeley, 1996.
[18] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," tech. rep., Graduate School of Industrial Administration, CMU, 1976.
[19] "Time-constrained data harvesting in wireless sensor networks." https://www.dropbox.com/s/lqq1ah9xec7m7e8/techreport.pdf.
[20] A. Blum, S. Chawla, D. Karger, T. Lane, A. Meyerson, and M. Minkoff, "Approximation algorithms for orienteering and discounted-reward tsp," *SIAM J. Comp.*, vol. 37, no. 2, pp. 653–670, 2007.
[21] P.-J. Wan, K. Alzoubi, and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," in *Proc. INFOCOM*, 2002.
[22] W. Whitt, "The efficiency of one long run versus independent replications in steady-state simulation," *Management Science*, vol. 37, no. 6, pp. 645–666, 1991.