Algorithmica (2018) 80:3158–3176
https://doi.org/10.1007/s00453-017-0373-6

Algorithms for Communication Scheduling in Data Gathering Network with Data Compression

Wenchang Luo[1,2] · Yao Xu[2] · Boyuan Gu[2] · Weitian Tong[3] · Randy Goebel[2] · Guohui Lin[2]

Received: 8 April 2016 / Accepted: 7 September 2017 / Published online: 12 September 2017

Abstract We consider a communication scheduling problem that arises within wireless sensor networks, where data is accumulated by the sensors and transferred directly to a central base station. One may choose to compress the data collected by a sensor, to decrease the data size for transmission, but the cost of compression must be considered. The goal is to designate a subset of sensors to compress their collected data, and then to determine a data transmission order for all the sensors, such that the total compression cost is minimized subject to a bounded data transmission completion time (a.k.a. makespan). A recent result confirms the NP-hardness for this problem, even in the special case where data compression is free. Here we first design a pseudo-polynomial time exact algorithm, articulated within a dynamic programming scheme. This algorithm also solves a variant with the complementary optimization goal—to

✉ Guohui Lin
guohui@ualberta.ca

Wenchang Luo
wenchang@ualberta.ca

Yao Xu
xu2@ualberta.ca

Boyuan Gu
bgu1@ualberta.ca

Weitian Tong
wtong@georgiasouthern.edu

Randy Goebel
rgoebel@ualberta.ca

1 Faculty of Science, Ningbo University, Ningbo 315211, Zhejiang, China

2 Department of Computing Science, University of Alberta, Edmonton, AB T6G 2E8, Canada

3 Department of Computer Sciences, Georgia Southern University, Statesboro, GA 30460, USA

minimize the makespan while constraining the total compression cost within a given budget. Our second result consists of a bi-factor $(1+\epsilon, 2)$-approximation for the problem, where $(1+\epsilon)$ refers to the compression cost and 2 refers to the makespan, and a 2-approximation for the variant. Lastly, we apply a sparsing technique to the dynamic programming exact algorithm, to achieve a dual fully polynomial time approximation scheme for the problem and a usual fully polynomial time approximation scheme for the variant.

Keywords Wireless sensor network · Data compression · Scheduling · Approximation algorithm · FPTAS · Dual FPTAS

1 Introduction

Data gathering wireless sensor networks (WSNs) have become an important framework for collecting data, and have found various applications in environment monitoring, surveillance and other areas [1]. In the general setting, a data gathering WSN consists of a set of sensors for data acquisition, and a base station. Because the remote wireless sensors have both limited power and computing capacity, the data collected by the sensors are to be transferred to the base station for storage and processing. For practical context, typical ratios of computation power to data transmission power are in the range of 4400 operations per byte transmission. It is well known that the efficiency of the whole network can be improved by careful communications management, through some specific communication protocols [8,15,19,22] and/or effective data gathering scheduling algorithms [2,3,7,17,18]. Besides these careful management details, another popular approach for improving the network performance is to compress the collected data, thus to decrease the data transmission time between the sensors and the base station. There is rich research on compression algorithm design and analysis for data gathering WSNs [6,14,16,21,23,24]. While compressing the collected data decreases the transmission time, there are at least two consequent challenges for network management. First, data compression consumes energy; second, data compression takes time, and delays data transmission. To address these issues, a theoretical model of data gathering networks with data compression has been recently formulated and studied by Berlińska [4], with the goal of minimizing the monetary (or energy) cost associated with data compression, subject to transferring all data (including original and compressed) directly to the base station within a given time bound.

1.1 Problem Description

We continue the study on the theoretical model pioneered in [4], from the approximation algorithm perspective; we also study a variant with the complementary optimization goal: to minimize the data transmission completion time (which is referred to as *makespan*, in the sequel) subject to a budget constraint on the total data compression cost. We next present the precise problem formulations, adopting the notations from [4].

In our data gathering WSN, there are a set of m identical sensors $\mathcal{P} = \{P_1, P_2, \ldots, P_m\}$ and a single base station denoted as S. All the data collected by the sensors, either in the original form or the compressed form, must be transferred directly to the base station; each data transmission should be done non-preemptively and no two sensors are allowed to transmit simultaneously. The size of data gathered by the sensor P_i is α_i, for $i = 1, 2, \ldots, m$; if one chooses to compress this data point (which will be denoted as α_i in the sequel), the compression algorithm decreases its size from α_i to $\gamma\alpha_i$, where $0 < \gamma < 1$. Compressing the data point α_i requires a monetary (or energy) cost of $f\alpha_i$, and the associated compression time is $A\alpha_i$, for some non-negative constants f and A. At time 0, the data gathered at any sensor P_i is ready for transmission; if the data point α_i is chosen for compression, the sensor P_i can start the data transmission only after the compression is fully done, that is, at or after time $A\alpha_i$. For data transmission time, although all the sensors are identical, due to location difference the sensor P_i needs time C_i for transferring one unit of data to the base station; these parameters C_i's are determined by the sensor locations and could be different from each other. The following theoretical model DG- Compr- OptF is defined in [4].

Problem 1 *(*DG- Compr- OptF [4]*).* Given m identical sensors $(P_i)_{1\le i\le m}$, and their parameters $(\alpha_i)_{1\le i\le m}$, γ, f, A, $(C_i)_{1\le i\le m}$ and a rational number T, find a subset of data points to compress such that the data compression cost is minimized and the makespan is within time T.

We define DG- Compr- OptT as the complementary problem to DG- Compr- OptF, as follows:

Problem 2 *(*DG- Compr- OptT*).* Given m identical sensors $(P_i)_{1\le i\le m}$, and their parameters $(\alpha_i)_{1\le i\le m}$, γ, f, A, $(C_i)_{1\le i\le m}$ and a rational number F, find a subset of data points to compress such that the data compression cost is within F and the makespan is minimized.

1.2 Preliminaries

It is not difficult to see that the central challenge in the above two problems, DG- Compr- OptF and DG- Compr- OptT, is to identify an optimal subset of sensors for compressing their gathered data. This is based on the following observation:

Observation 1 *If a subset $\mathcal{C} \subseteq \mathcal{P}$ of sensors for data compression is given, then not only (1) the compression cost is determined, but also (2) the makespan is determined.*

This observation holds because the target problem reduces to "scheduling on a single machine with job release times to minimize the makespan" (denoted as "$1|r_j|C_{\max}$"), for which sorting the jobs by non-decreasing release time gives an optimal schedule [5].

In our problem setting, a data point α_i has its release time 0 if non-compressed, or otherwise $A\alpha_i$. We therefore assume, without loss of generality, that the data points are always sorted in the non-decreasing size:

$$\alpha_1 \le \alpha_2 \le \ldots \le \alpha_m. \tag{1.1}$$

For a feasible solution in which the subset $\mathcal{C} \subseteq \mathcal{P}$ of data points are compressed, its compression cost is

$$f \sum_{i \in \mathcal{C}} \alpha_i. \tag{1.2}$$

Consider the optimal schedule obtained by sorting the data points by non-decreasing release time. If in this schedule the data transmission is continuous (i.e., the base station does not idle), then the makespan, denoted as $C_{\max}$, is

$$C_{\max} = \sum_{i \notin \mathcal{C}} C_i \alpha_i + \sum_{i \in \mathcal{C}} C_i \gamma \alpha_i; \tag{1.3}$$

otherwise the base station is idle while waiting for some data point of $\mathcal{C}$ to finish compression, then

$$C_{\max} = A\alpha_{i'} + \sum_{i \in \mathcal{C}, i \geq i'} C_i \gamma \alpha_i, \tag{1.4}$$

for some $i' \in \mathcal{C}$ (in fact, here the data point $\alpha_{i'}$ is the last one for which the base station has to wait for its compression to complete).

From Eqs. (1.2, 1.3, 1.4), and the fact that all the parameters (including the time bound T and the cost bound F) are rational numbers, we may assume without loss of generality that f is either 0 or 1 (representing whether compression is free or not, respectively), A is a non-negative integer, C_i for every i is a non-negative integer, and further that $\gamma \alpha_i$ and α_i for every i are non-negative integers, and lastly both T and F are non-negative integers too.

Berlińska [4] proved that both the problem DG- Compr- OptF and the variant DG- Compr- OptT are NP-hard even in the special case where $f = 0$ and all $\alpha_i = \alpha$ and in the special case where $A = 0$ and all $C_i = C$. Both reductions are from the well-known Partitioning problem [9]. The author also presented an exact algorithm to enumerate all possible subsets of data points for compression, of running time $O(2^m m \log m)$. To the best of our knowledge, no approximation algorithm has been reported for either DG- Compr- OptF or the variant DG- Compr- OptT.

1.3 Our Contributions

Recall the integrality assumption we made on the problem parameters. From Eq. (1.1) we know that α_m is the largest-size data point and we denote $\alpha = \alpha_m$; we denote $C = \max_{1 \leq i \leq m} C_i$. We first present a dynamic programming algorithm for the problem DG- Compr- OptF and the variant DG- Compr- OptT, and show that its running time is in $O(m^4 C^2 \alpha^3)$. This pseudo-polynomial time exact algorithm for both problems shows that they are weakly NP-hard. Our second result consists of a bi-factor $(1+\epsilon, 2)$-approximation algorithm for the problem DG- Compr- OptF, where $(1+\epsilon)$ refers to the compression cost and 2 refers to the makespan, and a 2-approximation algorithm for the variant DG- Compr- OptT. Lastly, based on the above two approximation algorithms, we present a sparsing technique on the dynamic programming algorithm, to derive a *dual* fully polynomial time approximation scheme [10] for the problem

DG- Compr- OptF, which returns a schedule of compression cost no greater than the optimum by possibly exceeding the given time bound by a factor of ϵ, for any $\epsilon > 0$; and to derive a usual fully polynomial time approximation scheme for the problem DG- Compr- OptT, that returns a schedule of makespan within $(1 + \epsilon)$ of the minimum, for any $\epsilon > 0$.

2 The Dynamic Programming Algorithm

The exact algorithm based on dynamic programming is designed for both the problem DG- Compr- OptF and the variant DG- Compr- OptT, which differs only in either the time bound T is given or the compression budget F is given. Essentially, our algorithm uses the data point order in Eq. (1.1) to examine whether or not to compress the data point α_j, for $j = 1, 2, \ldots, m$. The key to avoid running into exponential $O(2^m)$-time is not to record whether or not each data point α_j is compressed, but to record the resulting total compression cost, total data transmission time, and total idle time for the base station. To this purpose, we define a quadruple

$$(j; \ell_j, t_j, F_j)$$

with $j = 0, 1, \ldots, m$, for describing a typical *state*, which represents a partial schedule on the first j data points resulting in a total of ℓ_j units of base station idle time, a total of t_j units of data transmission time between the first j sensors and the base station, and a total compression cost of fF_j. Note that the base station never idles whenever there is a data point, original or compressed, ready for transmission. Therefore the makespan of this partial schedule is $\ell_j + t_j$. We start with the quadruple $(0; \ell_0, t_0, F_0) = (0; 0, 0, 0)$, representing a valid empty schedule.

We next describe how to obtain a partial schedule on the first $j + 1$ data points by adding the data points α_{j+1} to $(j; \ell_j, t_j, F_j)$, for $j = 0, 1, \ldots, m-1$, and considering both cases of whether or not to compress α_{j+1}. If the data point α_{j+1} is not compressed, then the total compression cost does not change; the data point α_{j+1} is ready for transmission at time 0 and it requires $C_{j+1}\alpha_{j+1}$ units of time for transmission. Clearly, if the previous total idle time $\ell_j > C_{j+1}\alpha_{j+1}$, then $C_{j+1}\alpha_{j+1}$ units can be used for transmission α_{j+1}; otherwise, all idle time is used up for transmission α_{j+1}. This gives rise to a state $(j + 1; \ell_{j+1}, t_{j+1}, F_{j+1})$

$$= (j + 1; \max\{\ell_j - C_{j+1}\alpha_{j+1}, 0\}, t_j + C_{j+1}\alpha_{j+1}, F_j).$$

On the other hand, if the data point α_{j+1} is chosen to be compressed, then the total compression cost increases by $f\alpha_{j+1}$; the data point α_{j+1} is ready for transmission at time $A\alpha_{j+1}$ and it requires $C_{j+1}\gamma\alpha_{j+1}$ units of time for transmission. Clearly, if the previous makespan $\ell_j + t_j < A\alpha_{j+1}$, then α_{j+1} starts transmission at time $A\alpha_{j+1}$, increasing the total idle time to $\ell_j + (A\alpha_{j+1} - \ell_j - t_j) = A\alpha_{j+1} - t_j$; otherwise, the earliest time for α_{j+1} to start transmission is time $\ell_j + t_j$, leaving the total idle time unchanged. This gives rise to a state $(j + 1; \ell_{j+1}, t_{j+1}, F_{j+1})$

$$= (j+1; \max\{A\alpha_{j+1} - t_j, \ell_j\}, t_j + C_{j+1}\gamma\alpha_{j+1}, F_j + \alpha_{j+1}).$$

Our dynamic programming (DP) algorithm uses a 4-dimensional binary table $DP(j; \ell_j, t_j, F_j)$ for computation, where the entry $(j; \ell_j, t_j, F_j)$ has a value 1 if and only if it represents a partial schedule on the first j data points resulting in a total of ℓ_j units of base station idle time, a total of t_j units of data transmission time between the first j sensors and the base station, and a total compression cost of fF_j. The recurrence for DP table entry computation is as follows:

$$DP(j+1; \ell_{j+1}, t_{j+1}, F_{j+1}) = \begin{cases} 1, & \text{if } \begin{cases} DP(j; \ell_j, t_j, F_j) = 1, \\ \ell_{j+1} = \max\{\ell_j - C_{j+1}\alpha_{j+1}, 0\}, \\ t_{j+1} = t_j + C_{j+1}\alpha_{j+1}, \\ F_{j+1} = F_j \end{cases} \\ 1, & \text{if } \begin{cases} DP(j; \ell_j, t_j, F_j) = 1, \\ \ell_{j+1} = \max\{A\alpha_{j+1} - t_j, \ell_j\}, \\ t_{j+1} = t_j + C_{j+1}\gamma\alpha_{j+1}, \\ F_{j+1} = F_j + \alpha_{j+1} \end{cases} \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

For initialization, $DP(0; 0, 0, 0) = 1$ and every other $(0; \ell_0, t_0, F_0)$ entry has a value 0. Since each entry $(j; \ell_j, t_j, F_j)$ is used for filling at most 2 other entries, the total computation time is linear in the size of the 4-dimensional table. Clearly, the first dimension has length $m+1$; the second and the third dimensions have length no more than $mC\alpha$, since for each j, $\ell_j + t_j$ is the makespan of the partial schedule which must be less than or equal to $\sum_{i=1}^{m} C_i\alpha_i \le mC\alpha$, corresponding to no data compression at all. The last dimension has length at most $m\alpha$. Therefore, the size of the DP table is $O(m^4C^2\alpha^3)$.

For the problem DG- Compr- OptF, where a time bound T is given, if $T \ge \sum_{i=1}^{m} C_i\alpha_i$, then the optimal schedule is not to compress any data point and thus to achieve the minimum compression cost 0; otherwise, the DP algorithm maintains a table of size only $O(m^2\alpha T^2)$ since the second and the third dimensions has length T. If no entry of the form $(m; \ell_m, t_m, F_m)$, such that $\ell_m + t_m \le T$, has a value 1, then there is no feasible solution to the problem; otherwise, from all those feasible solutions, the one having the minimum value in the F_m-field represents an optimal schedule. Scanning for this optimal schedule (or deciding that there is no feasible solution) can be done in $O(m\alpha T^2)$-time.

Likewise, for the variant DG- Compr- OptT, where a compression budget F is given, if $F \le f\sum_{i=1}^{m} \alpha_i$, then the DP algorithm maintains a table of size only $O(m^3C^2\alpha^2F)$ since the last dimension has length F. If no entry of the form $(m; \ell_m, t_m, F_m)$ has a value 1, then there is no feasible solution to the problem; otherwise, from all those feasible solutions, the one having the minimum sum $(\ell_m + t_m)$ represents an optimal schedule. Scanning for this optimal schedule (or deciding that there is no feasible solution) can be done in $O(m^2C^2\alpha^2F)$-time.

Theorem 1 *The problem* DG- Compr- OptF *admits an* $O(m^2\alpha T^2)$*-time exact algorithm based on dynamic programming, and the variant* DG- Compr- OptT *admits*

an $O(m^3C^2\alpha^2F)$-time exact algorithm based on dynamic programming, where $C = \max_{1\le i\le m} C_i$ and $\alpha = \max_{1\le i\le m} \alpha_i$. The running time for both algorithms is pseudo-polynomial, and thus the two problems are weakly NP-hard.

3 Approximating the Problem DG- Compr- OptF

In the problem DG- Compr- OptF, we may assume, without loss of generality, that for the sensor P_j,

$$A\alpha_j + C_j\gamma\alpha_j \le T, \text{ for any } j, \tag{3.1}$$

since otherwise the data point α_j cannot be compressed in any feasible solution and consequently can be excluded from compression.

We still assume the m sensors are indexed in the order specified by Eq. (1.1). Clearly, if $\sum_{j=1}^{m} C_j\alpha_j \le T$, then no data point has to be compressed and a trivial optimal schedule can be achieved with zero compression cost. The non-trivial case is that $\sum_{j=1}^{m} C_j\alpha_j > T$, where in any feasible schedule some data points have to be compressed. Our approximation algorithm is denoted as Approx1, and constructs a subset $\mathcal{C}$ of data points to be compressed by reducing the problem into the minimization Knapsack problem [12].

For this reduction, we define a binary variable x_j for the data point α_j, which has a value 1 if and only if α_j is packed into the knapsack $\mathcal{C}$. For each data point α_j, its weight is $C_j\alpha_j$ while its cost is α_j. The minimization Knapsack problem is to find a subset of data points such that the total weight is greater than or equal to some given constant bound (which is set to $(\sum_{j=1}^{m} C_j\alpha_j - T)/(1-\gamma)$ in our formulation) while the total cost is minimized, as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{m} \alpha_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{m} C_j\alpha_j x_j \ge \frac{\sum_{j=1}^{m} C_j\alpha_j - T}{1-\gamma}; \\ & x_j \in \{0,1\},\ j = 1,2,\ldots,m. \end{aligned} \tag{3.2}$$

Given any $\epsilon > 0$, the algorithm Approx1 invokes the FPTAS for the minimization Knapsack problem [12,20] to obtain a subset $\mathcal{C}$ of data points to be compressed, such that

$$\sum_{j\in\mathcal{C}} \alpha_j \le (1+\epsilon)\ \text{OPT}, \tag{3.3}$$

where OPT denotes the cost of the optimal knapsack to the minimization Knapsack problem in Eq. (3.2). Afterwards, the algorithm Approx1 determines the schedule by sorting the data points with non-decreasing release time, and the achieved makespan is denoted as $C_{\max}$.

Theorem 2 *Given any $\epsilon > 0$,* Approx1 *is a $(1+\epsilon, 2)$-approximation algorithm for the problem* DG- Compr- OptF*, with running time in $O(m/\epsilon + m\log m)$, where $(1+\epsilon)$ refers to the compression cost and 2 refers to the makespan.*

Proof We assume there are feasible solutions to the problem DG- COMPR- OPTF,[1] and let $\mathcal{C}^*$ denote the subset of data points to be compressed in the optimal solution, where $\mathcal{C}^* = \{\alpha_{j_1}, \alpha_{j_2}, \ldots, \alpha_{j_r}\}$ $(1 \leq j_1 < j_2 < \ldots < j_r \leq m)$. We distinguish two cases, corresponding to whether or not the base station ever idles in the optimal solution.

Case 1 *The base station idles in the optimal solution* Clearly, the base station only idles when it is waiting for a data point of $\mathcal{C}^*$ to finish the compression. Let ℓ_{j_i} denote the length of idle time associated with the data point $\alpha_{j_i} \in \mathcal{C}^*$ (which equals the starting transmission time for the data point α_{j_i}, minus the transmission completion time for the data point $\alpha_{j_{i-1}}$), for $i = 1, 2, \ldots, r$. The makespan of the optimal schedule is thus

$$C^*_{\max} = \sum_{j \notin \mathcal{C}^*} C_j \alpha_j + \sum_{j \in \mathcal{C}^*} \ell_j + \gamma \sum_{j \in \mathcal{C}^*} C_j \alpha_j. \tag{3.4}$$

Case 2 *The base station does not idle in the optimal solution* From Eq. (1.3), the makespan of the optimal schedule is

$$C^*_{\max} = \sum_{j \notin \mathcal{C}^*} C_j \alpha_j + \gamma \sum_{j \in \mathcal{C}^*} C_j \alpha_j. \tag{3.5}$$

In either of Case 1 and Case 2, we must have $C^*_{\max} \leq T$ and thus from Eqs. (3.4, 3.5) we have

$$\sum_{j \notin \mathcal{C}^*} C_j \alpha_j + \gamma \sum_{j \in \mathcal{C}^*} C_j \alpha_j \leq T.$$

Using $0 < \gamma < 1$, this leads to

$$\sum_{j \in \mathcal{C}^*} C_j \alpha_j \geq \frac{\sum_{j=1}^{m} C_j \alpha_j - T}{1 - \gamma}, \tag{3.6}$$

suggesting that the optimal solution is a feasible solution to the minimization KNAPSACK problem in Eq. (3.2). It follows from Eq. (3.3) that the compression cost of the solution by APPROX1 is

$$f \sum_{j \in \mathcal{C}} \alpha_j \leq f(1+\epsilon)\mathrm{OPT} \leq (1+\epsilon) f \sum_{j \in \mathcal{C}^*} \alpha_j,$$

that is, within $(1+\epsilon)$ of the optimum.

[1] The NP-hardness result by Berlińska [4], cited in the Preliminaries, hints that deciding whether or not there is a feasible solution to the problem DG- COMPR- OPTF is NP-complete.

Regarding the makespan of the computed schedule, since it is a feasible solution to the minimization KNAPSACK problem in Eq. (3.2), we have

$$\sum_{j\in\mathcal{C}} C_j\alpha_j \geq \frac{\sum_{j=1}^{m} C_j\alpha_j - T}{1-\gamma},$$

which is equivalent to

$$\sum_{j\notin\mathcal{C}} C_j\alpha_j + \gamma \sum_{j\in\mathcal{C}} C_j\alpha_j \leq T, \tag{3.7}$$

and Eq. (3.7) suggests that if the base station does not idle in the schedule, then by Eq. (1.3) the makespan $C_{\max} \leq T$.

If the base station idles in the computed schedule, then by Eqs. (1.4) and (3.1, 3.7) the makespan

$$C_{\max} = A\alpha_{i'} + \sum_{i\in\mathcal{C}, i\geq i'} C_i\gamma\alpha_i \leq T + T = 2T,$$

that is, within twice the given time bound T.[2]

In summary, the algorithm APPROX1 is a bi-factor $(1+\epsilon, 2)$-approximation for the problem DG- COMPR- OPTF. The running time of APPROX1 is dominated by invoking the FPTAS for the minimization KNAPSACK problem, which needs $O(m/\epsilon)$-time [12]; after the subset $\mathcal{C}$ is returned, finalizing the schedule takes an extra $O(m \log m)$-time. This finishes the proof. □

4 Approximating the Problem DG- COMPR- OPTT

In the case of the variant DG- COMPR- OPTT, similarly as in the last section, we may assume without loss of generality that for the sensor P_j,

$$f\alpha_j \leq F, \text{ for any } j, \tag{4.1}$$

since otherwise the data point α_j cannot be compressed in any feasible solution and consequently can be excluded from compression.

We still assume the m sensors are indexed in the order specified by Eq. (1.1). Firstly, we choose not to compress any data point to obtain a feasible schedule, denoted as π_0, of zero compression cost and of makespan $C^0_{\max} = \sum_{j=1}^{m} C_j\alpha_j$, which is an upper bound on the optimal makespan $C^*_{\max}$:

$$C^*_{\max} \leq C^0_{\max} = \sum_{j=1}^{m} C_j\alpha_j. \tag{4.2}$$

[2] This upper bound of $2T$ on the makespan holds regardless of existence of a feasible solution to the problem DG- COMPR- OPTF.

Our approximation algorithm is denoted as APPROX2, and constructs for each index i a subset of data points $\mathcal{C}_i \subseteq \{\alpha_1, \alpha_2, \ldots, \alpha_i\}$ to be compressed, by reducing the problem into the maximization KNAPSACK problem [12].

Define a binary variable x_j for the data point α_j, for $j = 1, 2, \ldots, i-1$, which has a value 1 if and only if α_j is packed into the knapsack $\mathcal{C}_i$. For each data point α_j, its weight is α_j while its profit is $C_j\alpha_j$. The maximization KNAPSACK problem is to find a subset of data points such that the total weight is less than or equal to some given constant bound (which is set to $F/f - \alpha_i$ in our formulation) while the total profit is maximized, as follows:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{i-1} C_j\alpha_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{i-1} \alpha_j x_j \le F/f - \alpha_i; \\ & x_j \in \{0, 1\},\ j = 1, 2, \ldots, i-1. \end{aligned} \tag{4.3}$$

Given any $\epsilon \le \gamma$, APPROX2 invokes the FPTAS for the maximization KNAPSACK problem [11] to obtain a subset of data points $\mathcal{C} = \{\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_q}\}$ $(1 \le i_1 < i_2 < \ldots < i_q \le i-1)$, such that

$$\sum_{j\in\mathcal{C}} C_j\alpha_j \ge (1-\epsilon)\ \mathrm{OPT}_i, \tag{4.4}$$

where OPT_i denotes the profit of the optimal knapsack to the maximization KNAPSACK problem in Eq. (4.3). Afterwards, the algorithm APPROX2 sets the subset of data points to be compressed as $\mathcal{C}_i = \mathcal{C} + \{\alpha_i\}$, and then determines the schedule π_i by sorting the data points with non-decreasing release time, and the achieved makespan is denoted as $C^i_{\max}$. At the end, APPROX2 outputs the one among $\{\pi_0, \pi_1, \pi_2, \ldots, \pi_m\}$ with the minimum makespan as the solution to the problem DG- COMPR- OPTT, denoted as π and its makespan denoted as $C_{\max}$.

Theorem 3 APPROX2 *is a* 2*-approximation algorithm for the problem* DG- COMPR- OPTT*, with running time in* $O(m^2/\gamma + m^2 \log m)$*, where* $0 < \gamma < 1$ *is the given data compression ratio.*

Proof Recall a major difference between the two problems DG- COMPR- OPTF and DG- COMPR- OPTT—there are always feasible solutions to the latter. We first see that every schedule π_i, for $i = 0, 1, 2, \ldots, m$, is feasible for the problem DG- COMPR- OPTT, because of the weight constraint in the maximization KNAPSACK problem. Let $\mathcal{C}^*$ denote the subset of data points to be compressed in the optimal solution, where $\mathcal{C}^* = \{\alpha_{j_1}, \alpha_{j_2}, \ldots, \alpha_{j_r}\}$ $(1 \le j_1 < j_2 < \ldots < j_r \le m)$. If $\mathcal{C}^* = \emptyset$, then the algorithm APPROX2 obtains an optimal schedule π with its makespan $C_{\max} = C^0_{\max} = C^*_{\max}$.

We next discuss the non-empty case of $\mathcal{C}^*$, and let $i = j_r$, that is, the data point α_i is the largest-size data point to be compressed in the optimal solution. Let $\mathcal{C}^*_{-i} =$

$\mathcal{C}^* - \{\alpha_i\}$. Clearly, $\mathcal{C}^*_{-i}$ is a feasible solution to the maximization KNAPSACK problem in Eq. (4.3), and therefore

$$\sum_{j \in \mathcal{C}^*_{-i}} C_j \alpha_j \leq \mathrm{OPT}_i. \tag{4.5}$$

On the other hand, we have

$$C^*_{\max} \geq \max \left\{ A\alpha_i + \gamma C_i \alpha_i, \sum_{j \notin \mathcal{C}^*} C_j \alpha_j + \gamma \sum_{j \in \mathcal{C}^*} C_j \alpha_j \right\}. \tag{4.6}$$

In the following we similarly distinguish two cases, corresponding to whether or not the base station ever idles in one of the computed solutions π_i. Recall that in π_i, the subset of compressed data points is $\mathcal{C}_i = \{\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_q}, \alpha_i\}$, where $1 \leq i_1 < i_2 < \ldots < i_q \leq i-1$. Let i_{q+1} denote i.

Case 1 *The base station idles in the computed solution* π_i Let i_k be the largest index of data points of $\mathcal{C}_i$ for which the base station idles and waits for its compression to be done. Then by Eqs. (1.1, 1.4) and (4.6) we have

$$\begin{aligned} C^i_{\max} &= A\alpha_{i_k} + \gamma \textstyle\sum_{j=k}^{q+1} C_{i_j} \alpha_{i_j} \\ &\leq A\alpha_i + \gamma C_i \alpha_i + \gamma \textstyle\sum_{j=k}^{q} C_{i_j} \alpha_{i_j} \\ &\leq C^*_{\max} + \gamma \textstyle\sum_{j=1}^{m} C_j \alpha_j \\ &\leq C^*_{\max} + C^*_{\max} \\ &= 2C^*_{\max}. \end{aligned}$$

Case 2 *The base station does not idle in the computed solution* π_i By Eqs. (1.1, 1.3) and (4.4, 4.5, 4.6), and using $\epsilon \leq \gamma$, we have

$$\begin{aligned} C^i_{\max} &= \textstyle\sum_{j \notin \mathcal{C}_i} C_j \alpha_j + \gamma \sum_{j \in \mathcal{C}} C_j \alpha_j + \gamma C_i \alpha_i \\ &= \textstyle\sum_{j=1}^{m} C_j \alpha_j - \sum_{j \in \mathcal{C}} C_j \alpha_j - C_i \alpha_i + \gamma \sum_{j \in \mathcal{C}} C_j \alpha_j + \gamma C_i \alpha_i \\ &\leq \textstyle\sum_{j=1}^{m} C_j \alpha_j - (1-\epsilon) \sum_{j \in \mathcal{C}^*_{-i}} C_j \alpha_j - C_i \alpha_i + \gamma \sum_{j \in \mathcal{C}} C_j \alpha_j + \gamma C_i \alpha_i \\ &= \textstyle\sum_{j=1}^{m} C_j \alpha_j - \sum_{j \in \mathcal{C}^*} C_j \alpha_j + \epsilon \sum_{j \in \mathcal{C}^*_{-i}} C_j \alpha_j + \gamma \sum_{j \in \mathcal{C}} C_j \alpha_j + \gamma C_i \alpha_i \\ &= \textstyle\sum_{j \notin \mathcal{C}^*} C_j \alpha_j + \epsilon \sum_{j \in \mathcal{C}^*_{-i}} C_j \alpha_j + \gamma \sum_{j \in \mathcal{C}} C_j \alpha_j + \gamma C_i \alpha_i \\ &\leq \textstyle\sum_{j \notin \mathcal{C}^*} C_j \alpha_j + \gamma \sum_{j \in \mathcal{C}^*} C_j \alpha_j + \gamma \sum_{j \in \mathcal{C}} C_j \alpha_j \\ &\leq C^*_{\max} + \gamma \textstyle\sum_{j=1}^{m} C_j \alpha_j \\ &\leq C^*_{\max} + C^*_{\max} \\ &= 2C^*_{\max}. \end{aligned}$$

In either of Case 1 and Case 2, we have $C^i_{\max} \leq 2C^*_{\max}$. It follows that the makespan of the computed solution by the algorithm APPROX2 is $C_{\max} \leq C^i_{\max} \leq 2C^*_{\max}$. The running time of one iteration of APPROX2 is dominated by invoking the FPTAS for the maximization KNAPSACK problem, which needs $O(m/\epsilon)$-time [11]; after the subset $\mathcal{C}$ is returned, finalizing the schedule takes an extra $O(m \log m)$-time. Note that we may set $\epsilon = \gamma$, and there are in total m iterations in the algorithm. This finishes the proof. □

5 Approximation Schemes

In this section, we begin with the pseudo-polynomial time exact algorithm based on dynamic programming in Sect. 2, to derive a *dual* fully polynomial time approximation scheme (FPTAS) [10] for the problem DG- COMPR- OPTF. By "dual", we mean that such an FPTAS has a twist, in that given any $\epsilon > 0$, it achieves a schedule of which the compression cost is no greater than the optimum, but the makespan might exceed the given time bound T by a factor ϵ. Its running time is polynomial in m and $\frac{1}{\epsilon}$. The key technique is sparsing [13], which essentially fills only a tiny fraction of the DP table in the exact algorithm. A normal FPTAS for the variant DG- COMPR- OPTT can be similarly derived, and its details are omitted here.

5.1 Algorithmic Setup

Recall that deciding whether or not there is a feasible solution to the problem DG- COMPR- OPTF is NP-complete. We assume there are feasible solutions to the problem DG- COMPR- OPTF, and subsequently denote the optimal schedule as π^*. We also assume the non-trivial case where $\sum_{j=1}^{m} C_j \alpha_j > T$, that is, in any feasible solution some data points have to be compressed.

Notice that if the largest-size data point to be compressed is known to be α_i, then all data points α_j with $j > i$ are to be transferred to the base station before any compressed data points. That is, any data point α_ℓ with $\ell \leq i$, when ready, can start transmission at or after time $\sum_{j=i+1}^{m} C_j \alpha_j$. Since we may try out this largest-size data point to be compressed by enumerating all possibilities, we assume in the sequel that the data point α_m is compressed in the optimal schedule π^*, and all data points can start transmission after time 0 as long as they are ready. We drop the unit compression cost f in the following to simplify the presentation. Then the compression cost of π^* is in between $\alpha = \alpha_m$ and $m\alpha$.

Recall that a quadruple

$$(j; \ell_j, t_j, F_j)$$

is defined in the dynamic programming algorithm for describing a *state*, which represents a partial schedule on the first j data points (see Eq. 1.1) resulting in a total of ℓ_j units of base station idle time, a total of t_j units of data transmission time between the first j sensors and the base station, and a total compression cost of F_j. Note that by time $A\alpha_m$, all data points, original or compressed, are either transferred or ready for transmission, we therefore have $\ell_j \leq A\alpha_m$ ($\leq T$). Also, from Theorem 2, the algorithm APPROX1 guarantees to output a schedule with its makespan $C_{\max} \leq 2T$; we thus may limit ourselves to consider only those states with $t_j \leq 2T$. Therefore, the range for each dimension of the quadruple can be bounded as follows:

$$\begin{aligned} &0 \leq j \leq m, \\ &0 \leq \ell_j \leq T, \\ &0 \leq t_j \leq 2T, \end{aligned} \tag{5.1}$$

$$0 \leq F_j \leq m\alpha.$$

5.2 The FPTAS

We are now ready to present our approximation scheme, denoted as APPROX3. Given any $\epsilon > 0$, define

$$\delta_1 = \frac{\epsilon}{2m}T,\ \delta_2 = \delta_1,\ \text{and}\ \delta_3 = \frac{\epsilon}{m}\alpha;$$

and set

$$v_1 = \lceil T/\delta_1 \rceil,\ v_2 = \lceil 2T/\delta_2 \rceil,\ \text{and}\ v_3 = \lceil m\alpha/\delta_3 \rceil.$$

We partition the range of the second dimension $[0, T]$ into v_1 intervals $I_1^\ell = [0, \delta_1]$, $I_i^\ell = ((i-1)\delta_1, i\delta_1]$, for $2 \leq i \leq v_1 - 1$, and $I_{v_1}^\ell = ((v_1-1)\delta_1, T]$; partition the range of the third dimension $[0, 2T]$ into v_2 intervals $I_1^t = [0, \delta_2]$, $I_i^t = ((i-1)\delta_2, i\delta_2]$, for $2 \leq i \leq v_2 - 1$, and $I_{v_2}^t = ((v_2-1)\delta_2, 2T]$; partition the range of the fourth dimension $[0, m\alpha]$ into v_3 intervals $I_1^F = [0, \delta_3]$, $I_i^F = ((i-1)\delta_3, i\delta_3]$, for $2 \leq i \leq v_3 - 1$, and $I_{v_3}^F = ((v_3 - 1)\delta_3, m\alpha]$.

Let $[n] = \{1, 2, \ldots, n\}$ for every positive integer n.

Given a triple $(q_1, q_2, q_3) \in [v_1] \times [v_2] \times [v_3]$, a box $B(j; q_1, q_2, q_3)$ is defined as the collection of the DP-table entries $(j; \ell_j, t_j, F_j)$ such that $\ell_j \in I_{q_1}^\ell$, $t_j \in I_{q_2}^t$, and $F_j \in I_{q_3}^F$. Our approximation scheme APPROX3 is also a dynamic programming, as in Sect. 2, and uses a much smaller 4-dimensional[3] binary table $DP'(j; q_1, q_2, q_3)$ for computation, where $0 \leq j \leq m$ and $(q_1, q_2, q_3) \in [v_1] \times [v_2] \times [v_3]$. The entry $DP'(j; q_1, q_2, q_3)$ has a value 1 if and only if there is a partial schedule on the first j data points resulting in a total base station idle time in $I_{q_1}^\ell$, a total data transmission time in $I_{q_2}^t$, and a total compression cost in $I_{q_3}^F$; associated with each such entry is the partial schedule with the least compression cost F_j (that is, all other partial schedules evaluating this DP' entry to 1 are discarded—ties broken arbitrarily).

For initialization, we have $DP'(0; 1, 1, 1) = 1$ and the associated partial schedule is $(0; 0, 0, 0)$; $DP'(0; q_1, q_2, q_3) = 0$, for all other triples $(q_1, q_2, q_3) \in [v_1] \times [v_2] \times [v_3]$.

Given an entry $DP'(j; q_1, q_2, q_3) = 1$ and its associated partial schedule $(j; \ell_j, t_j, F_j)$, we can derive two partial schedules $(j+1; \ell_{j+1}, t_{j+1}, F_{j+1})$ using the recurrence in Eq. (2.1), corresponding to compressing the data point α_{j+1} and not compressing α_{j+1}, respectively. Assuming ℓ_{j+1} (t_{j+1}, F_{j+1}, respectively) belongs to the interval $I_{p_1}^\ell$ ($I_{p_2}^t$, $I_{p_3}^F$, respectively), such a partial schedule $(j+1; \ell_{j+1}, t_{j+1}, F_{j+1})$ evaluates the entry $DP'(j+1; p_1, p_2, p_3)$ to 1, if the entry was previously 0; in this case, the partial schedule $(j; \ell_j, t_j, F_j)$ associated with the entry $DP'(j; q_1, q_2, q_3)$

[3] One could probably further reduce the size and/or the dimensionality of the table. We nonetheless use the current table for the ease of presentation.

is referred to as the *predecessor* of the partial schedule $(j+1; \ell_{j+1}, t_{j+1}, F_{j+1})$ associated with $DP'(j+1; p_1, p_2, p_3)$.

If the entry $DP'(j+1; p_1, p_2, p_3)$ was previously 1, then F_{j+1} is compared against the compression cost of the old partial schedule associated with the entry, and the one with a smaller value is kept (the predecessor information is correspondingly updated, if a change occurs). One key property of this tabular computation is that, for every saved partial schedule on the first $j+1$ data points, its predecessor partial schedule on the first j data points is also saved.

At the end of filling the DP' table, for each triple $(q_1, q_2, q_3) \in [v_1] \times [v_2] \times [v_3]$ such that $DP'(m; q_1, q_2, q_3) = 1$, we examine the associated schedule $(m; \ell_m, t_m, F_m)$: if the makespan $\ell_m + t_m \leq (1+\epsilon)T$, then it is considered as a candidate schedule; from all candidate schedules, our approximation scheme APPROX3 outputs the one with the minimum compression cost, denoted as $(m; \ell_m^\epsilon, t_m^\epsilon, F_m^\epsilon)$, as the final solution to the problem DG- COMPR- OPTF.

Theorem 4 *For any $\epsilon > 0$, the problem* DG- COMPR- OPTF *admits an $O(m^6/\epsilon^3)$-time approximation algorithm based on dynamic programming, which returns a schedule of compression cost no more than the optimum, by possibly exceeding the given time bound by a factor of ϵ.*

Proof Recall that we assume the existence of the optimal schedule π^* and the data point α_m is compressed in π^*. We drop the unit compression cost f in the following to simplify the presentation. Using the state notation, let $\pi^* = (m; \ell_m^*, t_m^*, F_m^*)$; then we have $0 \leq \ell_m^* + t_m^* \leq T$ and $\alpha \leq F_m^* \leq m\alpha$.

Let the following path of states computed by the exact algorithm for the problem DG- COMPR- OPTF in Sect. 2 denote how the state $(0; 0, 0, 0)$ leads to the optimal schedule $\pi^* = (m; \ell_m^*, t_m^*, F_m^*)$:

$$\begin{aligned}(0;0,0,0) &\to (1; \ell_1^*, t_1^*, F_1^*)\\ &\to (2; \ell_2^*, t_2^*, F_2^*)\\ &\to \dots\\ &\to (m-1; \ell_{m-1}^*, t_{m-1}^*, F_{m-1}^*) \to (m; \ell_m^*, t_m^*, F_m^*).\end{aligned} \tag{5.2}$$

We prove the following claim by induction:

Claim 1 *For each $j \in [m]$, there is a partial schedule $(j; \ell_j, t_j, F_j)$ saved by the approximation algorithm* APPROX3*, such that $|\ell_j - \ell_j^*| \leq j\delta_1$, $|t_j - t_j^*| \leq j\delta_2$, and $F_j \leq F_j^*$.*

Base Case ($j = 1$) During the computation of the algorithm APPROX3, two partial schedules are generated and they are $(1; 0, C_1\alpha_1, 0)$ and $(1; A\alpha_1, \gamma C_1\alpha_1, f\alpha_1)$ (corresponding to whether or not the data point α_1 is compressed or not, respectively). If they don't belong to the same box $B(1; \cdot, \cdot, \cdot)$, then both of them are saved by APPROX3; otherwise only $(1; 0, C_1\alpha_1, 0)$ is saved due to its smaller compression cost. Since $(1; \ell_1^*, t_1^*, F_1^*)$ must be either of them, in the former case the claim holds, as there is a saved partial schedule identical to $(1; \ell_1^*, t_1^*, F_1^*)$; in the latter case, the claim holds because the saved partial schedule $(1; 0, C_1\alpha_1, 0)$ and $(1; \ell_1^*, t_1^*, F_1^*)$ belong to the same box $B(1; \cdot, \cdot, \cdot)$.

Inductive Step We assume that for each $i \in [j]$, there is a partial schedule $(i; \ell_i, t_i, F_i)$ saved by the approximation algorithm APPROX3, such that

$$|\ell_i - \ell_i^*| \le i\delta_1,\ |t_i - t_i^*| \le i\delta_2,\ \text{and } F_i \le F_i^*.$$

We distinguish two cases corresponding to whether or not the data point α_{j+1} is compressed in the optimal schedule π^*, that is, how the state $(j; \ell_j^*, t_j^*, F_j^*)$ leads to $(j+1; \ell_{j+1}^*, t_{j+1}^*, F_{j+1}^*)$.

Case 1 *The data point α_{j+1} is not compressed in the optimal schedule π^** In this case, from Eq. (2.1), we have

$$\ell_{j+1}^* = \max\{\ell_j^* - C_{j+1}\alpha_{j+1}, 0\},\ t_{j+1}^* = t_j^* + C_{j+1}\alpha_{j+1},\ \text{and } F_{j+1}^* = F_j^*.$$

We examine the partial schedule $(j+1; \ell'_{j+1}, t'_{j+1}, F'_{j+1})$ generated from $(j; \ell_j, t_j, F_j)$ by not compressing the data point α_{j+1}, which by Eq. (2.1) satisfies

$$\ell'_{j+1} = \max\{\ell_j - C_{j+1}\alpha_{j+1}, 0\},\ t'_{j+1} = t_j + C_{j+1}\alpha_{j+1},\ \text{and } F'_{j+1} = F_j.$$

It follows that

$$\begin{aligned}
|\ell'_{j+1} - \ell_{j+1}^*| &= |\max\{\ell_j - C_{j+1}\alpha_{j+1}, 0\} - \max\{\ell_j^* - C_{j+1}\alpha_{j+1}, 0\}| \\
&\le \max\{|\ell_j - \ell_j^*|, \max\{\ell_j - \ell_j^*, 0\}, \max\{\ell_j^* - \ell_j, 0\}\} \\
&= |\ell_j - \ell_j^*| \\
&\le j\delta_1; \\
|t'_{j+1} - t_{j+1}^*| &= |t_j - t_j^*| \le j\delta_2,\ \text{and } F'_{j+1} = F_j \le F_j^* = F_{j+1}^*.
\end{aligned}$$

Case 2 *the data point α_{j+1} is compressed in the optimal schedule π^** In this case, from Eq. (2.1), we have

$$\ell_{j+1}^* = \max\{A\alpha_{j+1} - t_j^*, \ell_j^*\},\ t_{j+1}^* = t_j^* + C_{j+1}\gamma\alpha_{j+1},\ \text{and } F_{j+1}^* = F_j^* + \alpha_{j+1}.$$

We examine the partial schedule $(j+1; \ell'_{j+1}, t'_{j+1}, F'_{j+1})$ generated from $(j; \ell_j, t_j, F_j)$ by compressing the data point α_{j+1}, which by Eq. (2.1) satisfies

$$\ell'_{j+1} = \max\{A\alpha_{j+1} - t_j, \ell_j\},\ t'_{j+1} = t_j + C_{j+1}\gamma\alpha_{j+1},\ \text{and } F'_{j+1} = F_j + \alpha_{j+1}.$$

It follows that

$$\begin{aligned}
|\ell'_{j+1} - \ell_{j+1}^*| &= |\max\{A\alpha_{j+1} - t_j^*, \ell_j^*\} - \max\{A\alpha_{j+1} - t_j, \ell_j\}| \\
&\le \max\{|t_j - t_j^*|, |\min\{t_j^* - t_j, \ell_j^* - \ell_j\}|, |\max\{t_j - t_j^*, \ell_j^* - \ell_j\}|\} \\
&\le \max\{|t_j - t_j^*|, |\ell_j - \ell_j^*|\} \\
&\le j\delta_1 \text{ (due to } \delta_1 = \delta_2); \\
|t'_{j+1} - t_{j+1}^*| &= |t_j - t_j^*| \le j\delta_2,\ \text{and } F'_{j+1} = F_j + \alpha_{j+1} \le F_j^* + \alpha_{j+1} = F_{j+1}^*.
\end{aligned}$$

Note that in either of the above two cases, the examined partial schedule $(j+1; \ell'_{j+1}, t'_{j+1}, F'_{j+1})$ might not be saved by the algorithm Approx3, due to its compression cost F'_{j+1} not being the least among the partial schedules residing in the same box $B(j+1; \cdot, \cdot, \cdot)$. Let the partial schedule associated with the same box $B(j+1; \cdot, \cdot, \cdot)$ be $(j+1; \ell_{j+1}, t_{j+1}, F_{j+1})$, which is saved by Approx3. We have

$$|\ell_{j+1} - \ell'_{j+1}| \leq \delta_1, \ |t_{j+1} - t'_{j+1}| \leq \delta_2, \text{ and } F_{j+1} \leq F'_{j+1}.$$

Therefore, we conclude that for the saved partial schedule $(j+1; \ell_{j+1}, t_{j+1}, F_{j+1})$,

$$|\ell_{j+1} - \ell^*_{j+1}| \leq (j+1)\delta_1, \ |t_{j+1} - t^*_{j+1}| \leq (j+1)\delta_2, \text{ and } F_{j+1} \leq F^*_{j+1}.$$

This finishes the proof of the claim.

From the above Claim 1, the algorithm Approx3 saves a schedule $(m; \ell_m, t_m, F_m)$, for which

$$|\ell_m - \ell^*_m| \leq m\delta_1, \ |t_m - t^*_m| \leq m\delta_2, \text{ and } F_m \leq F^*_m.$$

Consequently,

$$(\ell_m + t_m) \leq (\ell^*_m + t^*_m) + m\delta_1 + m\delta_2 \leq T + \epsilon T = (1+\epsilon)T,$$

and

$$F_m \leq F^*_m = \text{OPT},$$

where $\text{OPT} = F^*_m$ is the minimum compression cost.

Recall that the algorithm Approx3 examines all saved complete schedules $(m; \ell'_m, t'_m, F'_m)$ and they are considered candidates if and only if the makespan $\ell'_m + t'_m \leq (1+\epsilon)T$. From Claim 1, we conclude that the candidate list is non-empty, and from this list Approx3 outputs the one $(m; \ell^\epsilon_m, t^\epsilon_m, F^\epsilon_m)$ with the minimum compression cost, which is certainly no more than OPT. This finishes the proof of the performance ratio of the algorithm Approx3.

For the running time, note that the size of the DP' table is $O(m \times v_1 \times v_2 \times v_3) = O(m \times \frac{m}{\epsilon} \times \frac{m}{\epsilon} \times \frac{m^2}{\epsilon}) = O(\frac{m^5}{\epsilon^3})$. Each table entry is associated with at most one partial schedule, which leads to at most two partial schedules on an additional data point. That is, the total tabular computation time, as well as saving the associated partial schedules, is in $O(\frac{m^5}{\epsilon^3})$. Afterwards, a non-dominant amount of time is spent for finding the solution. Note that the above presentation of the algorithm Approx3 assumes the data point α_m is compressed in the optimal schedule π^*. To remove this assumption, we may execute the algorithm m times, each time assuming the largest-size data point being compressed in the optimal schedule is α_j. This results in a total running time of Approx3 in $O(\frac{m^6}{\epsilon^3})$. □

We state the following similar result on the variant DG- Compr- OptT, with only a sketch of proof for the (better) running time as it is similar to the proof of Theorem 4.

Theorem 5 *For any $\epsilon > 0$, the problem* DG- COMPR- OPTT *admits an $O(m^4/\epsilon^3)$-time $(1+\epsilon)$-approximation algorithm based on dynamic programming.*

Proof Again we drop the unit compression cost f in the following to simplify the presentation.

For the problem DG- COMPR- OPTT, we can develop the same approximation scheme as APPROX3 based on dynamic programming and the sparsing technique. The only difference is the size of the DP' table, which becomes $O(m \times v_1 \times v_2 \times v_3) = O(m \times \frac{m}{\epsilon} \times \frac{m}{\epsilon} \times \frac{m}{\epsilon}) = O(\frac{m^4}{\epsilon^3})$ by defining

$$\delta_1 = \delta_2 = \frac{\epsilon}{2m} T \text{ and } \delta_3 = \frac{\epsilon}{m} F$$

and setting

$$v_1 = v_2 = \lceil T/\delta_1 \rceil \text{ and } v_3 = \lceil F/\delta_3 \rceil,$$

for any given $\epsilon > 0$, with T estimated using APPROX2 and F given as a part of input. Since we only execute the algorithm once, the total running time is thus in $O(\frac{m^4}{\epsilon^3})$. □

6 Concluding Remarks

We investigated a communication scheduling problem in data gathering wireless sensor networks with data compression. There are two objectives: to minimize the total data compression cost, and to minimize the makespan. Two single-objective optimization problems are defined with the other measure bounded by a given value, denoted as DG- COMPR- OPTF and DG- COMPR- OPTT, respectively. Both problems have been proven NP-hard; we presented the first pseudo-polynomial time exact algorithm based on dynamic programming, thus showing that they are (only) weakly NP-hard. Using this dynamic programming algorithm, we derived a dual fully polynomial time approximation scheme [10] for the problem DG- COMPR- OPTF, that returns a schedule of compression cost no greater than the optimum by possibly exceeding the given time bound by a factor of ϵ, for any $\epsilon > 0$; a usual fully polynomial time approximation scheme for the problem DG- COMPR- OPTT can also be derived, that returns a schedule of makespan within $(1+\epsilon)$ of the minimum, for any $\epsilon > 0$.

It is worth mentioning that deciding whether an instance of the problem DG- COMPR- OPTF has a feasible solution is already NP-complete; therefore our dual FPTAS is probably the best possible.

In the dynamic program algorithms, we intentionally separate the data transmission completion time into two parts, namely the total base station idle time ℓ and the total data transmission time t. Such a separation not only simplifies the derivation of the recurrence in Eq. (2.1), but also ensures Claim 1 in the proof of Theorem 4; as otherwise

the differences between the corresponding quantities might grow exponentially in j (currently they grow only linearly in j).

Acknowledgements W.L. was supported by China Scholarship Council (Grant No. 201408330402), the K. C. Wong Magna Fund in the Ningbo University, and the Ningbo Natural Science Foundation (2016A610078). W.L., Y.X., B.G., R.G. and G.L. were supported by NSERC. W.T. was supported by the FY16 Startup Funding from the Georgia Southern University.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Comput. Netw. **38**, 393–422 (2002)
2. Alfieri, A., Bianco, A., Brandimarte, P., Chiasserini, C.F.: Maximizing system lifetime in wireless sensor networks. Eur. J. Oper. Res. **181**, 390–402 (2007)
3. Berlińska, J.: Communication scheduling in data gathering networks with limited memory. Appl. Math. Comput. **235**, 530–537 (2014)
4. Berlińska, J.: Scheduling for data gathering networks with data compression. Eur. J. Oper. Res. **246**, 744–749 (2015)
5. Błażewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Weglarz, J.: Handbook on Scheduling: From Theory to Applications. Springer, Berlin (2007)
6. Cheng, J., Ye, Q., Jiang, H., Wang, D., Wang, C.: Stcdg: an efficient data gathering algorithm based on matrix completion for wireless sensor networks. IEEE Trans. Wirel. Commun. **12**, 850–861 (2013)
7. Choi, K., Robertazzi, T.G.: Divisible load scheduling in wireless sensor networks with information utility. In: Proceedings of the 2008 IEEE International Performance, Computing and Communications Conference, pp. 9–17 (2008)
8. Ergen, S.C., Varaiya, P.: TDMA scheduling algorithms for wireless sensor networks. Wirel. Netw. **16**, 985–997 (2010)
9. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. W. H. Freeman and Company, San Francisco (1979)
10. Hochbaum, D., Shmoys, D.: Using dual approximation algorithms for scheduling problems: theoretical and practical results. J. ACM **34**, 144–162 (1987)
11. Kellerer, H., Pferschy, U.: Improved dynamic programming in connection with an FPTAS for the knapsack problem. J. Comb. Optim. **8**, 5–11 (2004)
12. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Berlin (2004)
13. Kellerer, H., Strusevich, V.: Fast approximation schemes for Boolean programming and scheduling problems related to positive convex half-product. Eur. J. Oper. Res. **228**, 24–32 (2013)
14. Kimura, N., Latifi, S.: A survey on data compression in wireless sensor networks. In: Proceedings of the 2005 International Conference on Information Technology: Coding and Computing, pp. 8–13 (2005)
15. Kumar, S., Chauhan, S.: A survey on scheduling algorithms for wireless sensor networks. Int. J. Comput. Appl. **20**, 7–13 (2011)
16. Luo, C., Wu, F., Sun, J., Chen, C.W.: Compressive data gathering for large-scale wireless sensor networks. In: Proceedings of the 15th Annual International Conference on Mobile Computing and Networking, pp. 145–156 (2009)
17. Moges, M., Robertazzi, T.G.: Wireless sensor networks: scheduling for measurement and data reporting. IEEE Trans. Aerosp. Electron. Syst. **42**, 327–340 (2006)
18. Rossi, A., Singh, A., Sevaux, M.: Lifetime maximization in wireless directional sensor network. Eur. J. Oper. Res. **231**, 229–241 (2013)
19. Shi, L., Fapojuwo, A.O.: TDMA scheduling with optimized energy efficiency and minimum delay in clustered wireless sensor networks. IEEE Trans. Mob. Comput. **9**, 927–940 (2010)
20. Tauhidul, I.M.: Approximation Algorithms for Minimum Knapsack Problem. Master's thesis, University of Lethbridge (2009)
21. Wang, J., Tang, S., Yin, B., Li, X.Y.: Data gathering in wireless sensor networks through intelligent compressive sensing. In: INFOCOM 2012, pp. 603–611 (2012)

22. Wu, Y., Li, X.Y., Liu, Y., Lou, W.: Energy-efficient wake-up scheduling for data collection and aggregation. IEEE Trans. Parallel Distrib. Syst. **21**, 275–287 (2010)
23. Xiang, L., Luo, J., Rosenberg, C.: Compressed data aggregation: energy-efficient and high-fidelity data collection. IEEE/ACM Trans. Netw. **21**, 1722–1735 (2013)
24. Xu, L., Wang, Y., Wang, Y.: Major coefficients recovery: a compressed data gathering scheme for wireless sensor network. In: Global Telecommunications Conference (GLOBECOM 2011), pp. 1–5 (2011)