Algorithmica
DOI 10.1007/s00453-016-0171-6

Max-min Fair Rate Allocation and Routing in Energy Harvesting Networks: Algorithmic Analysis

Jelena Marašević[1] · **Clifford Stein**[2] · **Gil Zussman**[1]

Received: 29 May 2015 / Accepted: 30 May 2016

Abstract This paper considers max-min fair rate allocation and routing in energy harvesting networks where fairness is required among *both the nodes and the time slots*. Unlike most previous work on fairness, we focus on *multihop topologies* and consider *different routing methods*. We assume a predictable energy profile and focus on the design of efficient and optimal algorithms that can serve as benchmarks for distributed and approximate algorithms. We first develop an algorithm that obtains a max-min fair rate assignment for any routing that is specified at the input. We then turn to the problem of determining a "good" routing. For *time-invariable unsplittable routing*, we develop an algorithm that finds routes that maximize the minimum rate assigned to any node in any slot. For *fractional routing*, we derive a combinatorial algorithm for the *time-invariable* case with constant rates. We show that the *time-variable* case is at least as hard as the 2-commodity feasible flow problem and design an FPTAS to

A partial and preliminary version of this paper appeared in Proc. ACM MobiHoc'14. This research was supported in part by NSF Grants CCF-1349602, CCF-1421161, CCF-09-64497, and CNS-10-54856 and the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA Grant Agreement No. [PIIF-GA-2013-629740].11.

✉ Jelena Marašević
jelena@ee.columbia.edu

Clifford Stein
cliff@ieor.columbia.edu

Gil Zussman
gil@ee.columbia.edu

1 Department of Electrical Engineering, Columbia University,
500 W. 120th St., 1300 S. W. Mudd, New York, NY 10027, USA

2 Department of Industrial Engineering and Operations Research, Columbia University,
500 W. 120th St., 300 S. W. Mudd, New York, NY 10027, USA

combat the high running time. Finally, we show that finding an unsplittable routing or a routing tree that provides lexicographically maximum rate assignment (i.e., the best in the max-min fairness terms) is NP-hard, even for a time horizon of a single slot. Our analysis provides insights into the problem structure and can be applied to other related fairness problems.

Keywords Energy harvesting · Energy adaptive networking · Network flows · Sensor networks · Routing · Fairness

1 Introduction

Recent advances in the development of ultra-low-power transceivers and energy harvesting devices (e.g., solar cells) will enable self-sustainable and perpetual wireless networks [11,14,15]. In contrast to legacy wireless sensor networks, where the available energy only decreases as the nodes sense and forward data, in energy harvesting networks the available energy can also increase through a replenishment process. This added energy replenishment results in significantly more complex variations of the available energy, which poses challenges in the design of resource allocation and routing algorithms.

The problems of resource allocation, scheduling, and routing in energy harvesting networks have received considerable attention [2,4,9,12,13,16–18,23,24,29,32,34]. Most existing work considers simple networks consisting of a single node or a link [2,4,13,16,29,34]. Moreover, fair rate assignment has not been thoroughly studied, and most of the work either focuses on maximizing the total (or average) throughput [2,4,9,12,18,23,27,29,32,34], or considers fairness either *only over nodes* [24] or *only over time* [13,16]. An exception is [17], which requires fairness over both the nodes and the time, but is limited to *two nodes*.

In this paper, we study the max-min fair rate assignment and routing problems for general network topologies, requiring fairness over both nodes and time slots, and with the goal of designing optimal and efficient algorithms.

Following [9,13,16,17,23,24], we assume that the harvested energy is known for each node over a finite time horizon T. Such a setting corresponds to a highly-predictable energy profile, and can also be used as a benchmark for evaluating algorithms designed for unpredictable energy profiles. We consider an energy harvesting sensor network with a single sink node, and network connectivity modeled by a directed graph (Fig. 1). Each node senses some data from its surrounding (e.g., air pressure, temperature, radiation level), and sends it to the sink. The nodes spend their energy on sensing, sending, and receiving data.

1.1 Fairness Motivation

Two natural conditions that a network should satisfy are:

(i) balanced data acquisition across the entire network, and

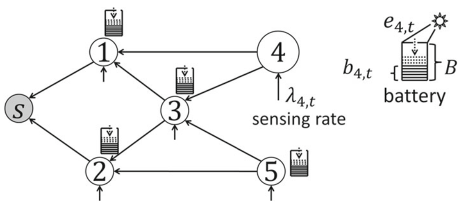


Fig. 1 A simple energy harvesting network: the nodes sense the environment and forward the data to a sink s. Each node has a battery of capacity B. At time t node i's battery level is $b_{i,t}$, it harvests $e_{i,t}$ units of energy, and senses at data rate $\lambda_{i,t}$

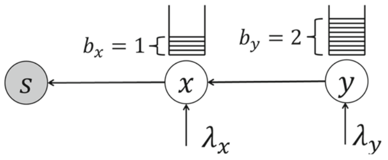


Fig. 2 An example of a network in which throughput maximization can result in a very unfair rate allocation among the nodes

(ii) persistent operation (i.e., even when the environmental energy is not available for harvesting).

Condition (i) is commonly maintained by requiring fairness of the sensing rates over the nodes in each time slot. We note that in the considered network model, due to different energy costs for sending, sensing, and receiving data, throughput maximization can be inherently unfair even in the case of single-slot time horizon. For example, consider a simple network with two energy harvesting nodes x and y and a sink s as illustrated in Fig. 2. Assume that x has one unit of energy available, and y has two units of energy. Let c_{st} denote the joint cost of sensing and sending a unit flow, and let c_{rt} denote the joint cost for receiving and sending a unit flow. Let λ_x and λ_y denote the sensing rates assigned to the nodes x and y, respectively. Suppose that the objective is to maximize $\lambda_x + \lambda_y$. If $c_{\mathrm{st}} = 1$, $c_{\mathrm{rt}} = 2$, then in the optimal solution $\lambda_x = 1$ and $\lambda_y = 0$. Conversely, if $c_{\mathrm{st}} = 2$, $c_{\mathrm{rt}} = 1$, then in the optimal solution $\lambda_x = 0$ and $\lambda_y = 1$. This example easily extends to more general degenerate cases in which maximum-throughput solution assigns non-zero sensing rates only to one part of the network, whereas the remaining nodes do not send any data to the sink.

One approach to achieving (ii) is by assigning constant sensing rates to the nodes. However, this approach can result in underutilization of the available energy. As a simple example, consider a node that harvests outdoor light energy over a 24-h time horizon. If the battery capacity is small, then the sensing rate must be low to prevent battery depletion during the nighttime. However, during the day-time, when the harvesting rates are high, a low sensing rate prevents full utilization of the energy that can be harvested. Therefore, it is advantageous to vary the sensing rates over time. However, fairness must be required over time slots to prevent the rate assignment algorithm from assigning high rates during periods of

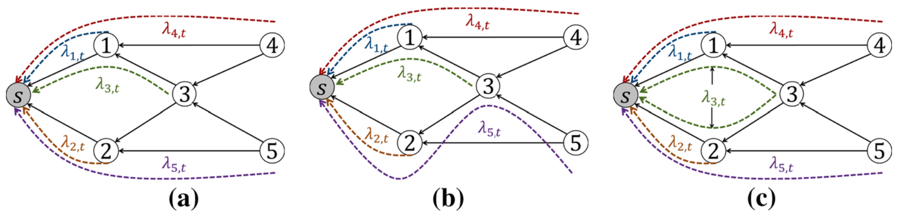


Fig. 3 Routing types: **a** a routing tree, **b** unsplittable routing: each node sends its data over one path, **c** fractional routing: nodes can send their data over multiple paths. Paths are represented by *dashed lines*

high energy availability, and zero rates when no energy is available for harvesting.

To guarantee (i) and (ii), we seek a lexicographically maximum rate assignment $\Lambda = \{\lambda_{i,t}\}$, where $i \in \{1, \ldots, n\}$ indexes nodes, while $t \in \{1, \ldots, T\}$ indexes time slots. Informally, a rate assignment $\Lambda = \{\lambda_{i,t}\}$ is lexicographically maximum if it is feasible, and for any alternative rate assignment $\Lambda' = \{\lambda'_{i,t}\}$, by traversing the elements of Λ and Λ' in non-decreasing order either all the elements from Λ and Λ' are equal, or in the first pair of non-equal elements the greater element is from Λ. Such a lexicographically maximum rate assignment is equivalent to the most egalitarian rate assignment—namely, the max-min fair rate assignment—whenever a max-min fair rate assignment exists. Formal definitions of max-min fairness and lexicographical ordering of vectors are provided in Sect. 2.1.

1.2 Routing Types

We consider three different routing types that can be used in any fixed time slot: (i) a routing tree, (ii) unsplittable (single-path) routing, and (iii) fractional (multi-path routing), illustrated in Fig. 3. During one time slot, the routing and the assigned rates are fixed.

A routing tree is the simplest routing: every node i (except for the sink) has a single parent node to which it sends all the flow that i either generates through sensing or receives from other nodes. Unsplittable (or single-path) routing is a generalization of the routing tree, where every node has a single path to the sink over which it sends all the flow it generates. Finally, fractional (or multi-path) routing is the most general form of routing in which every node can split the generated flow over arbitrarily many paths to the sink.

The routing tree is a special case of the unsplittable routing, and the unsplittable routing is a special case of the fractional routing. Therefore, it is clear that (under any reasonable comparison criteria) on any input graph out of the three routing types the routing trees support the "lowest" rates, while the fractional routings support the "highest" rates. We illustrate the effect of the routing type on the minimum rate assigned in a max-min fair rate assignment in Figs. 4 and 5.

We will refer to a routing as *time-invariable*, if in every time slot each node i uses the same set of paths to send its flow to the sink, and, moreover, for each path used

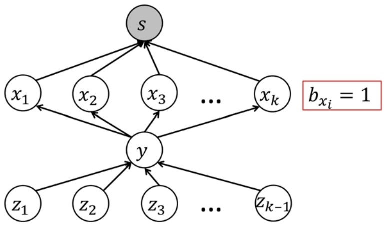


Fig. 4 A network example in which unsplittable routing provides minimum sensing rate that is $\Omega(n)$ times higher than for any routing tree. Assume $c_{\mathrm{st}} = c_{\mathrm{rt}} = 1$ and $T = 1$. Available energy levels at all the nodes x_i, $i \in \{1, \ldots, k\}$ are equal to 1, as shown in the *box* next to the nodes. Other nodes have energy levels that are high enough so that they are not constraining. In any routing tree, y has some x_i as its parent, so $\lambda_{x_i} = \lambda_y = \lambda_{z_1} = \cdots = \lambda_{z_{k-1}} = 1/(k+1)$ and $\lambda_{x_j} = 1$ for $j \neq i$. In an unsplittable routing with paths $p_{x_i} = \{x_i, s\}$, $p_{z_i} = \{z_i, y, x_i, s\}$, and $p_y = \{y, x_k, s\}$, all the rates are equal to 1/2. As $k = \Theta(n)$, the minimum rate improves by $\Omega((k+1)/2) = \Omega(n)$

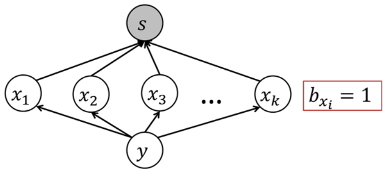
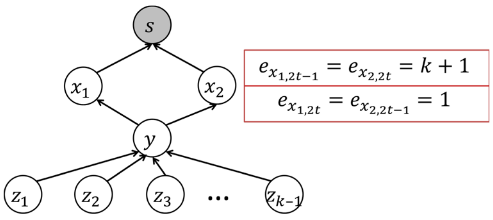


Fig. 5 A network example in which a fractional routing provides minimum sensing rate that is $\frac{2}{1+1/(n-1)} \approx$ 2 times higher than in any unsplittable routing. Assume $c_{\mathrm{st}} = c_{\mathrm{rt}} = 1$ and $T = 1$. Available energy levels at all the nodes are equal to 1, as shown in the *box* next to the nodes. In any unsplittable routing, y sends all its flow through one x_i, so $\lambda_{x_i} = \lambda_y = \frac{1}{2}$ and $\lambda_{x_j} = 1$ for $j \neq i$. In a fractional routing, y can split its flow over all x_i's, so that $\frac{\lambda_y}{n-1} + \lambda_{x_i} = b_{x_i}$, for all i. To maximize minimum assigned rate, $\lambda_y = \lambda_{x_i} = \frac{1}{1+1/(n-1)}$. Therefore, the minimum assigned rate improves by a factor of $\frac{2}{1+1/(n-1)}$

by i the *fraction* of flow sent by i does not change over time slots.[1] Otherwise, the routing is *time-variable*. For example, we will say that a routing is a time-variable routing tree, if the most complex routing used in any time slot is a routing tree. As any time-invariable routing is a special case of the corresponding time-variable routing, the time-variable routings in general provide higher rates. We illustrate the effect of the time variance of a routing on the minimum rate assigned to any node in a max-min fair rate assignment in Fig. 6.

It is natural to ask why should any simpler routing type be preferred over time-variable fractional routing—the most general one. The answer lies in the practical implementation of a routing: in general, more complex routing types are more difficult to maintain and require more control information that consumes energy thus effectively lowering the achievable sensing rates [15].

[1] Note that node i's sensing rate (generated flow) can change over time, even though the routing does not change.



s
x_1
x_2
y
z_1
z_2
z_3
$\cdots$
z_{k-1}
$e_{x_{1,2t-1}} = e_{x_{2,2t}} = k+1$
$e_{x_{1,2t}} = e_{x_{2,2t-1}} = 1$

Fig. 6 A network example in which a time-variable routing solution provides minimum sensing rate that is $\Omega(n)$ times higher than in any time-invariable routing. The batteries of x_1 and x_2 are initially empty, and the battery capacity at all the nodes is $B = 1$. Harvested energy values over time slots for nodes x_1 and x_2 are shown in the *box* next to them. Other nodes are assumed not to be energy constraining. In any time-invariable routing, at least one of x_1, x_2 has $\Omega(k) = \Omega(n)$ descendants, forcing its rate to the value of $1/\Omega(n)$ in the slots in which the harvested energy value is equal to 1. In a routing in which y sends the data only through x_1 in odd slots and only through x_2 in even slots: $\lambda_y = \lambda_{z_1} = \cdots = \lambda_{z_{k-1}} = 1$

Table 1 Our results for determining a max-min fair routing

Routing	Computational complexity
Routing tree	NP-hard to approximate within $O(\log(n))$ even for $T = 1$
Unsplittable routing	NP-hard to determine even for $T = 1$
Time-variable fractional routing	Can be determined with an $\widetilde{O}(nT(T^2\epsilon^{-2} \cdot (nT + MCF(n,m) + LP(mT, nT)))$-time algorithm, where $MCF(n,m)$ is the running time of an algorithm that solves the min-cost flow problem on a graph with n nodes and m edges and $LP(mT, nT)$ is the running time of an algorithm that solves a linear program with mT variables and nT constraints
Time-invariable fractional routing with time-invariable rates	Can be determined with an $\widetilde{O}(n(T + MF(n,m)))$-time algorithm, where $MF(n,m)$ is the running time of an algorithm that solves the maximum flow problem on a graph with n nodes and m edges

1.3 Our Contributions

For a routing that is provided at the input, we design a combinatorial algorithm that solves the max-min fair rate assignment problem. The algorithm runs in $\widetilde{O}(nmT^2)$ time,[2] where n is the number of energy-harvesting nodes, m is the number of edges in the routing graph, and T is the time horizon.

We then turn to the problem of finding a “good” routing of the specified type, where a routing is “good” if it provides a lexicographically maximum rate assignment out of all feasible routings of the same type. We sometimes refer to such a routing as the max-min fair routing [3] (see Sect. 2.2 for a formal statement of the problems). Our results for determining a max-min fair routing of a specified type are summarized in Table 1.

We show that a **max-min fair routing tree** is NP-hard to approximate within $\Omega(\log(n))$ and that a **max-min fair unsplittable routing** is NP-hard to find, regard-

[2] $\widetilde{O}(.)$-notation hides poly-log terms.

[3] The notions of max-min fairness and lexicographical ordering of vectors are defined in Sect. 2.1.

less of whether the routing is time variable or not. Relaxing the requirement of the lexicographically maximum rates, we design a polynomial algorithm that determines a **time-invariable unsplittable routing that maximizes the minimum rate** assigned to any node in any time slot.

For the **max-min fair *time-variable* fractional routing**, we demonstrate that verifying whether a given rate assignment is feasible is at least as hard as solving a feasible 2-commodity flow. This result implies that, to our current knowledge, it is unlikely that we can determine a max-min fair fractional routing without the use of linear programming (LP). To combat the high running time induced by the LP, we develop a fully polynomial time approximation scheme (FPTAS). We also show that in the special case when the fractional routing is restricted to be ***time-invariable*** **with rates that are constant over time**, the max-min fair routing can be determined in polynomial time with a combinatorial algorithm that we provide in Sect. 5.

Our algorithms rely on the well-known water-filling framework, described in Sect. 2.1. It is important to note that water-filling is a framework—not an algorithm—and therefore it does not specify how to solve the maximization nor fixing of the rates steps (see Sect. 2.1). Even though a general LP framework for implementing water-filling such as e.g., [8,24,31] can be adapted to solve some of the problems studied in this paper, their implementation in general requires solving $O(N^2)$ LPs for any problem with N variables. For instance, to determine a max-min fair time-variable fractional routing this water filling framework would in general need to solve $O(n^2T^2)$ LPs with $O(mT)$ variables and $O(nT)$ constraints, thus resulting in an unacceptably high running time. Our algorithms are devised relying on the problem structure, and in most cases do not use LP. The only exception is the algorithm for determining a max-min fair time-variable fractional routing (Sect. 4), which solves $O(nT)$ LPs, and thus provides at least $O(nT)$-fold improvement as compared to an adaptation of [8,24,31].

The problems considered generalize classical max-min fair routing problems that have been studied outside the area of energy harvesting networks, such as: max-min fair fractional routing [28], max-min fair unsplittable routing [21], and bottleneck routing [3]. In contrast to the problems studied in [3,21,28], our model allows different costs for flow generation and forwarding, and has time-variable node capacities determined by the available energies at the nodes. We remark that studying networks with node capacities is as general as studying networks with capacitated edges, as there are standard methods for transforming one of these two problems into the other (see, e.g., [1]). Therefore, we believe that the results will find applications in other related areas.

1.4 Related Work

We briefly survey the related work on classical fairness problems and problems arising in sensor and energy-harvesting networking applications.

Energy-harvesting Networks Rate assignment in energy harvesting networks in the case of a single node or a link was studied in [2,4,9,13,16,29,34]. Resource allocation and scheduling for network-wide scenarios using the Lyapunov optimization technique was studied in [12,18,27,32]. While the work in [12,18,27,32] can support unpredictable energy profiles, it focuses on the (sum-utility of) time-average rates, which is, in general, time-unfair. Online algorithms for resource allocation and routing were considered in [10,23].

Max-min time-fair rate assignment for a single node or a link was considered in [13,16], while max-min fair energy allocation for single-hop and two-hop scenarios was studied in [17]. Similar to our work, [17] requires fairness over both the nodes and the time slots, but considers only two energy harvesting nodes. The work on max-min fairness in network-wide scenarios [24] is explained in more detail below.

Sensor Networks A special case of max-min fair rate assignment and routing in energy harvesting networks is related to the problems of lifetime maximization in sensor networks (see, e.g., [6,26] and the follow-up work). In particular, the problem of maximizing *only the minimum rate* assigned to any node (instead of finding a max-min fair rate assignment) over a time horizon of *a single slot* is equivalent to maximizing the lifetime of a sensor network.

Determining a maximum lifetime tree in sensor networks as in [5] is a special case of determining a max-min fair tree routing in energy harvesting networks. We extend the NP-hardness result from [5] and provide a lower bound of $\Omega(\log n)$ for the approximation ratio (for both [5] and our version of the problem), where n is the number of nodes in the network.

Max-min Fair Rate Assignment Max-min fair rate assignment for a given routing was studied extensively (see [3,7] and references therein). Max-min fair rate assignment in energy harvesting networks reduces to the problems from [3,7] for $c_{\mathrm{st}} = c_{\mathrm{rt}}$ (unit energy costs) and $T = 1$ (static capacities). In the energy harvesting network setting, the problem of rate assignment has been considered in [24], for rates that are constant over time and a time-invariable routing tree. We consider a more general case than in [24], where the rates are time-variable, fairness is required over both network nodes and time slots, and the routing can be time-variable and of any type (a routing tree, an unsplittable routing, or a fractional routing).

Max-min Fair Unsplittable Routing Determining a max-min fair unsplittable routing as studied in [21] is a special case of determining a max-min fair unsplittable routing in energy harvesting networks for $c_{\mathrm{st}} = c_{\mathrm{rt}}$ and $T = 1$. The NP-hardness results from [21] implies the NP-hardness of determining a max-min fair unsplittable routing in energy harvesting networks.

Max-min Fair Fractional Routing Max-min fair fractional routing was first studied in [28]. The algorithm from [28] relies on the property that the total values of a max-min fair flow and max flow are equal, which does not hold even in simple instances of energy harvesting networks. The problem of determining a max-min fair fractional routing reduces to the problem of [28] for $T = 1$ and $c_{\mathrm{st}} = c_{\mathrm{rt}}$.

Max-min fair fractional routing in energy harvesting networks has been considered in [24]. The distributed algorithm from [24] is a heuristic for the problem of determining a time-invariable fractional routing with constant rates. We provide a combinatorial algorithm that solves this problem optimally in a centralized manner (Sect. 5). We focus on the more general problem of determining a max-min fair time-variable routing with time-variable rates, and we provide an FPTAS for this problem in Sect. 4.

A general linear programming framework for max-min fair routing was provided in [31], and extended to the setting of sensor and energy harvesting networks in [8] and [24], respectively. This framework, when applied to our setting, is highly inefficient.

1.5 Organization of the Paper

The rest of the paper is organized as follows. Section 2 provides background on max-min fairness and lexicographic maximization, and introduces the model and the considered problems. Section 3 considers rate assignment in a given routing, while Sects. 4 and 5 study determining a max-min fair fractional routing in time-variable and time-invariable settings, respectively. Section 6 provides hardness results for determining unsplittable routing or a routing tree. Section 7 provides conclusions and outlines possible future directions.

2 Preliminaries

2.1 Max-min Fairness and Lexicographic Maximization

A vector v is max-min fair if it is feasible and no element v_i of v can be increased without either violating feasibility or decreasing some other element $v_j \le v_i$. A more formal definition of max-min fairness is (see, e.g., [3]):

Definition 1 An allocation vector $v = (v_1, \dots, v_l)$ is max-min fair if it is feasible, and for any other feasible vector $u = (u_1, \dots, u_l)$: if $u_i > v_i$ for some $i \in \{1, \dots, l\}$, then there exists $j \in \{1, \dots, l\}$ such that $v_j < v_i$ and $v_j > u_j$.

Closely related to the max-min fairness is the notion of lexicographic maximization. The lexicographic ordering of vectors, with the relational operators denoted by $\overset{lex}{=}$, $\overset{lex}{>}$, and $\overset{lex}{<}$, is defined as follows:

Definition 2 Let u and v be two vectors of the same length l, and let u_s and v_s denote the vectors obtained from u and v respectively by sorting their elements in the non-decreasing order. Then:

(i) $u \overset{lex}{=} v$ if $u_s = v_s$ element-wise;
(ii) $u \overset{lex}{>} v$ if there exists $j \in \{1, 2, \dots, l\}$, such that $u_s(j) > v_s(j)$, and $u_s(1) = v_s(1), \dots, u_s(j-1) = v_s(j-1)$ if $j > 1$;
(iii) $u \overset{lex}{<} v$ if neither $u \overset{lex}{=} v$ nor $u \overset{lex}{>} v$.

A max-min fair allocation vector exists on any convex and compact set [31]. In a given optimization problem whenever a max-min fair vector exists, it is unique and equal to the lexicographically maximum one [33]. We summarize these two results in the following lemma:

Lemma 1 *For any convex and compact feasible region, a max-min fair allocation vector exists and it is unique. Moreover, the max-min fair vector is equivalent to the lexicographically maximum vector from the same feasible region.*

Lexicographic maximization of a vector v over a feasible region $\mathcal{R}$ can be implemented using the water-filling framework (see, e.g., [3]):

Algorithm 1 WATER-FILLING-FRAMEWORK($\mathcal{R}$)

```
1: Set $v_i = 0\ \forall i$, and mark all the elements of $v$ as not fixed.
2: MAXIMIZING-THE-RATES: Increase all the elements $v_i$ of $v$ that are not fixed by the same maximum
      amount, subject to the constraints from $\mathcal{R}$.
3: FIXING-THE-RATES: Fix all the $v_i$'s that cannot be further increased.
4: If all the elements of $v$ are fixed, terminate. Otherwise, go to Step 2.
```

As we will see later, the problems of finding the max-min fair rate assignment in a given routing and determining max-min fair fractional routing will have convex and compact feasible regions. Since in this case max-min fair rate allocation is equivalent to the lexicographically maximum one (Lemma 1), our algorithms will rely on the WATER-FILLING-FRAMEWORK. The algorithmic challenges for these problems will lie in the efficient implementation of common rate maximization (Step 2) and rate fixing (Step 3).

For problems that do not have a convex feasible region, a max-min fair allocation does not necessarily exist, while a lexicographically maximum allocation always exists (see, e.g., [31]). Therefore, the problems of finding an "optimal" routing tree or an unsplittable routing may not have a solution in the max-min fair sense, but will always have at least one solution in the context of lexicographic maximization. For this reason, we will consider lexicographic maximization in such cases.

2.2 Model and Problem Formulation

We consider a network that consists of n energy harvesting nodes and one sink node (Fig. 1). The sink node is assumed not to be energy constrained. In the rest of the paper, we will use "sink" to refer to the sink node and "node" to refer to an energy harvesting node. The connectivity between the nodes is modeled by a directed graph $G = (V, E)$, where $|V| = n + 1$ (n nodes and the sink), and $|E| = m$. We assume without loss of generality that every node has a directed path to the sink, because otherwise it can be removed from the graph. The main notation is summarized in Table 2.

Each node is equipped with a rechargeable battery of finite capacity B. The time horizon is T time slots. The duration of a time slot is assumed to be much longer than the duration of a single data packet, but short enough so that the rate of energy harvesting does not change during a slot. For example, if outdoor light energy is harvested, one time slot can be at the order of a minute. In a time slot t, a node i harvests $e_{i,t}$ units of energy. The battery level of a node i at the beginning of a time slot t is $b_{i,t}$. We follow a predictable energy profile [9,13,16,17,23,24], and assume that all the values of harvested energy $e_{i,t}$, $i \in \{1, \ldots, n\}$, $t \in \{1, \ldots, T\}$, battery capacity B, and all the initial battery levels $b_{i,1}$, $i \in \{1, \ldots, n\}$ are known and finite.

A node i in slot t senses data (generates flow) at rate $\lambda_{i,t}$. A node forwards all the data it senses and receives towards the sink. The flow on a link (i, j) in slot t is denoted by $f_{ij,t}$. Each node spends c_{s} energy units to generate a unit flow, and c_{tx}, respectively

Table 2 Nomenclature

Inputs	
n	Number of energy harvesting nodes
m	Number of edges
T	Time horizon
i	Node index, $i \in \{1, 2, \dots n\}$
t	Time index, $t \in \{1, \dots, T\}$
B	Battery capacity
$e_{i,t}$	Harvested energy at node i in time slot t
c_{s}	Energy spent for sensing a unit flow
c_{tx}	Energy spent for transmitting a unit flow
c_{rx}	Energy spent for receiving a unit flow
Variables	
$\lambda_{i,t}$	Sensing rate of node i in time slot t
$f_{ij,t}$	Flow on link (i, j) in time slot t
$b_{i,t}$	Battery level at node i at the beginning of time slot t
Notation	
c_{st}	Energy spent for jointly sensing and transmitting a unit flow: $c_{\mathrm{st}} = c_{\mathrm{s}} + c_{\mathrm{tx}}$
c_{rt}	Energy spent for jointly receiving and transmitting a unit flow: $c_{\mathrm{rt}} = c_{\mathrm{rx}} + c_{\mathrm{tx}}$
$f_{i,t}^{\Sigma}$	Total flow entering node i in time slot t: $f_{i,t}^{\Sigma} = \sum_{j:(j,i)\in E} f_{ji,t}$

c_{rx}, energy units to send, respectively receive, a unit flow. The joint cost of generating and sending a unit flow is denoted by $c_{\mathrm{st}} \equiv c_{\mathrm{s}} + c_{\mathrm{tx}}$, while the joint cost of receiving and sending a unit flow is denoted by $c_{\mathrm{rt}} \equiv c_{\mathrm{rx}} + c_{\mathrm{tx}}$.

Consider any routing $\mathcal{R} = \{\mathcal{R}_t\}$, where $\mathcal{R}_t \subseteq E$ is a subset of edges from the underlying graph G used to route data in time slot t. The feasible region of the sensing rates $\lambda_{i,t}$ and the flows $f_{ij,t}$ with respect to a given routing $\mathcal{R}$ is determined by the following set of linear[4] constraints:

$$\forall i \in \{1, \dots, n\}, t \in \{1, \dots, T\}:$$

$$\sum_{(j,i)\in\mathcal{R}_t} f_{ji,t} + \lambda_{i,t} = \sum_{(i,j)\in\mathcal{R}_t} f_{ij,t} \tag{1}$$

$$b_{i,t+1} = \min\left\{B,\ b_{i,t} + e_{i,t} - \left(c_{\mathrm{st}}\lambda_{i,t} + c_{\mathrm{rt}} \sum_{(j,i)\in\mathcal{R}_t} f_{ji,t}\right)\right\} \tag{2}$$

$$b_{i,t+1} \ge 0,\ \lambda_{i,t} \ge 0,\ f_{ij,t} \ge 0, \forall (i, j) \in \mathcal{R}_t, \tag{3}$$

where (1) is a classical flow conservation constraint, while (2) describes the battery evolution over time slots.

[4] Note that we treat Eq. (2) as a linear constraint, since the considered problems focus on maximizing $\lambda_{i,t}$'s (under the max-min fairness criterion), and (2) can be replaced by $b_{i,t+1} \le B$ and $b_{i,t+1} \le b_{i,t} + e_{i,t} - (c_{\mathrm{rt}} f_{i,t}^{\Sigma} + c_{\mathrm{st}}\lambda_{i,t})$ while leading to the same solution.

For Definitions 1 and 2 to apply, we will interpret a rate assignment $\Lambda = \{\lambda_{i,t}\}$ as a one-dimensional vector.

2.3 Considered Problems

We examine different routing types, in time-variable and time-invariable settings, as described in the Introduction. The problems that we consider are from either of the following two categories: (i) determining a max-min fair rate assignment in a routing that is provided at the input, and (ii) determining a routing of the required type that provides lexicographically maximum rate assignment. We specify the problems in more detail below. The first problem formalizes (i), while the remaining problems are specific instances of (ii).

P-DETERMINE-RATES: Given a routing $\mathcal{R} = \{\mathcal{R}_{i,t}\}$, determine the max-min fair assignment of the rates $\{\lambda_{i,t}\}$. Note that this setting subsumes all the routing types that were defined in the Introduction.

P-UNSPLITTABLE-ROUTING: For a given (time-invariable or time-variable) unsplittable routing $\mathcal{P}$, let $\{\lambda_{i,t}^{\mathcal{P}}\}$ denote a rate allocation that optimally solves P-DETERMINE-RATES over $\mathcal{P}$. Searching over all feasible unsplittable routings in graph G over time horizon T, determine an unsplittable routing $\mathcal{P}$ that provides a lexicographically maximum assignment of rates $\{\lambda_{i,t}^{\mathcal{P}}\}$.

P-ROUTING-TREE: Let $\mathcal{T}$ denote a (time-invariable or time-variable) routing tree on the input graph G. For each $\mathcal{T}$, let $\{\lambda_{i,t}^{\mathcal{T}}\}$ denote a rate allocation that optimally solves P-DETERMINE-RATES. Searching over all feasible routing trees in G over time horizon T, determine $\mathcal{T}$ that provides a lexicographically maximum assignment of rates $\{\lambda_{i,t}^{\mathcal{T}}\}$.

P-FRACTIONAL-ROUTING: Determine a *time-variable* fractional routing that supports lexicographically maximum rate assignment $\{\lambda_{i,t}\}$, considering all the (time-variable, fractional) routings.

P-FIXED-FRACTIONAL-ROUTING: Determine a *time-invariable* fractional routing that provides the max-min fair time-invariable rate assignment $\{\lambda_{i,t}\} = \{\lambda_i\}$. This problem is a special case of P-FRACTIONAL-ROUTING, where the routing and the rates are constant over time.

3 Rate Allocation in a Specified Routing

This section provides an algorithm for P-DETERMINE-RATES, the problem of rate assignment for a routing specified at the input. The analysis applies to any routing type described in the Introduction. As discussed in Sect. 2.1, to design an efficient rate assignment algorithm relying on WATER-FILLING-FRAMEWORK, we need to implement the common rate maximization (Step 2) and fixing of the rates (Step 3) of WATER-FILLING-FRAMEWORK efficiently.

We begin by introducing additional notation. We assume that the routing over time $t \in \{1, \ldots, T\}$ is provided as a time-sequence of sets of routing paths $\mathcal{P} = \{\mathcal{P}_{i,t}\}$ from a node i to the sink s, for each node $i \in V \backslash \{s\}$. We also assume that associated with each path $p_{i,t} \in \mathcal{P}_{i,t}$ there is a coefficient $\alpha_{i,t} > 0$, such that $\sum_{p_{i,t} \in \mathcal{P}_{i,t}} \alpha_{i,t} = 1$. The coefficients $\alpha_{i,t}$ determine the fraction of flow $\lambda_{i,t}$ that is sent over path $p_{i,t}$. We say that node j is a descendant of node i in a time slot t if $i \in \mathcal{P}_{j,t}$, that is, if i is on at least one routing path of j in slot t.[5]

We let $F_{i,t}^{k} = 1$ if the rate $\lambda_{i,t}$ is not fixed at the beginning of the kth iteration of WATER-FILLING-FRAMEWORK, $F_{i,t}^{k} = 0$ otherwise. Initially, $F_{i,t}^{1} = 1, \forall i, t$. If a rate $\lambda_{i,t}$ is not fixed, we will say that it is "active". To concisely evaluate the flow incoming into node i in time slot t in iteration k, we let $D_{i,t}^{k} = \sum_{\{p_{j,t}: j \neq i \wedge i, p_{j,t} \in \mathcal{P}_{j,t}\}} \alpha_{j,t} \cdot F_{j,t}^{k}$. Finally, let $\lambda_{i,t}^{k}$ and $b_{i,t}^{k}$ denote the values of $\lambda_{i,t}$ and $b_{i,t}$ in the kth iteration of WATER-FILLING-FRAMEWORK, where $\lambda_{i,t}^{0} = 0, \forall i, t$. Under this notation, the rates in kth iteration can be expressed as $\lambda_{i,t}^{k} = \sum_{l=1}^{k} F_{i,t}^{l} \lambda^{l}$, where λ^{l} denotes the common amount by which all the active rates get increased in the lth iteration. Moreover, it is not hard to see that the total flow incoming into node i and originating at other nodes in iteration k is equal to $\sum_{l=1}^{k} D_{i,t}^{l} \lambda^{l}$.

3.1 Maximizing the Rates

Using the notation introduced in this section, maximization of the common rate λ^{k} in kth iteration of WATER-FILLING-FRAMEWORK can be formulated as follows:

$$\begin{aligned}
&\mathbf{max}\ \lambda^{k}\\
&\quad \mathbf{s.t.} \forall i \in \{1, \ldots, n\}, t \in \{1, \ldots, T\}:\\
&\qquad b_{i,t+1}^{k} = \min\{B,\ b_{i,t}^{k} + e_{i,t} - \sum_{l=1}^{k} \lambda^{l} (c_{\mathrm{rt}} D_{i,t}^{l} + c_{\mathrm{st}} F_{i,t}^{l})\}\\
&\qquad b_{i,t}^{k} \geq 0,\ \lambda^{k} \geq 0,
\end{aligned}$$

where $\forall i\ \forall k : b_{i,1}^{k} = b_{i,1}$.

Instead of using all of the λ^{l}'s from previous iterations in the expression for $b_{i,t+1}^{k}$, we can define the battery drop in the iteration k, for node i and time slot t as: $\Delta b_{i,t}^{k} = \sum_{l=1}^{k} \lambda^{l} \left(c_{\mathrm{rt}} D_{i,t}^{l} + c_{\mathrm{st}} F_{i,t}^{l}\right)$ and only keep track of the battery drops from the previous iteration. The intuition is as follows: to determine the battery levels in all the time slots, we only need to know the initial battery level and how much energy ($\Delta b_{i,t}$) is spent per time slot. Setting $\Delta b_{i,t}^{0} = 0$, the problem can be written as:

$$\begin{aligned}
&\mathbf{max}\ \lambda^{k}\\
&\quad \mathbf{s.t.} \forall i \in \{1, \ldots, n\}, t \in \{1, \ldots, T\}:\\
&\qquad \Delta b_{i,t}^{k} = \Delta b_{i,t}^{k-1} + \lambda^{k} \left(c_{\mathrm{rt}} D_{i,t}^{k} + c_{\mathrm{st}} F_{i,t}^{k}\right)\\
&\qquad b_{i,t+1}^{k} = \min\left\{B,\ b_{i,t}^{k} + e_{i,t} - \Delta b_{i,t}^{k}\right\}
\end{aligned}$$

5 Notice that this is consistent with the definition of a descendant in a routing tree.

$$b_{i,t}^k \geq 0,\ \lambda^k \geq 0$$

Writing the problem for each node independently, we can solve the following subproblem:

$$\textbf{max}\quad \lambda_i^k \tag{4}$$

$$\textbf{s.t.}\ \forall\ t \in \{1, \ldots, T\}:$$

$$\Delta b_{i,t}^k = \Delta b_{i,t}^{k-1} + \lambda_i^k \left(c_{\mathrm{rt}} D_{i,t}^k + c_{\mathrm{st}} F_{i,t}^k \right) \tag{5}$$

$$b_{i,t+1}^k = \min \left\{ B,\ b_{i,t}^k + e_{i,t} - \Delta b_{i,t}^k \right\} \tag{6}$$

$$b_{i,t}^k \geq 0,\ \lambda_i^k \geq 0 \tag{7}$$

for each i with $\sum_{i,t} F_{i,t}^k > 0$, and determine $\lambda^k = \min_i \lambda_i^k$. Notice that we can bound each λ_i^k by the interval $[0, \lambda_{\max,i}^k]$, where $\lambda_{\max,i}^k$ is the rate for which node i spends all its available energy in the first slot τ in which its rate is not fixed:

$$\lambda_{\max,i}^k = \frac{b_{i,\tau}^{k-1} + e_{i,\tau}}{c_{\mathrm{rt}} D_{i,\tau}^k + c_{\mathrm{st}}}, \quad \tau = \min\{t : F_{i,t}^k = 1\}.$$

The subproblem of determining λ_i^k can now be solved by performing a binary search in the interval $[0, \lambda_{\max,i}^k]$.

Let δ denote the precision of the input variables. Note that however small, δ can usually be expressed as a constant. This section can be summarized in the following lemma.

Lemma 2 MAXIMIZING-THE-RATES *in* P-DETERMINE-RATES *can be implemented in time*

$$O\Big(T \sum_i \log\Big(\frac{\lambda_{\max,i}^k}{\delta}\Big)\Big) = O\Big(nT \log\Big(\frac{B + \max_{i,t} e_{i,t}}{\delta c_{st}}\Big)\Big).$$

3.2 Fixing the Rates

Recall that the elements of the matrix F^k are such that $F_{i,t}^k = 0$ if the rate $\lambda_{i,t}$ is fixed for the iteration k, and $F_{i,t}^k = 1$ otherwise. At the end of iteration $k \geq 1$, let $F^{k+1} = F^k$, and consider the following set of fixing rules:

(F1) For all (i, t) such that $b_{i,t+1}^k = 0$ set $F_{i,t}^{k+1} = 0$.

(F2) For all (i, t) such that $b_{i,t+1}^k = 0$ determine the longest sequence $(i, t), (i, t-1), (i, t-2), \ldots, (i, \tau)$, $\tau \geq 1$, with the property that $b_{i,s}^k + e_{i,s} - \Delta b_{i,s}^k \leq B$ $\forall s \in \{t, t-1, \ldots, \tau\}$, and set $F_{i,s}^{k+1} = 0\ \forall s$.

(F3) For all (i, t) for which the rules (F1) and (F2) have set $F_{i,t}^{k+1} = 0$, and for all j such that $i \in \mathcal{P}_{j,t}$, set $F_{j,t}^{k+1} = 0$.

We will need to prove that these rules are necessary and sufficient for fixing the rates. Here, "necessary" means that no rate that gets fixed at the end of iteration k can get increased in iteration $k+1$ without violating at least one of the constraints. "Sufficient" means that all the rates $\lambda_{i,t}$ with $F_{i,t}^{k+1}=1$ can be increased by a positive amount in iteration $k+1$ without violating feasibility.

Lemma 3 (Necessity) *No rate fixed by the rules (F1), (F2) and (F3) can be increased in the next iteration without violating feasibility constraints.*

Proof We will prove the lemma by induction on iteration k.

The base case consider the first iteration and observe the pairs (i,t) for which $F_{i,t}^{1}=0$.

Suppose that $b_{i,t+1}^{1}=0$. The first iteration starts with all the rates being active, so we get from the constraint (6):

$$\begin{aligned}
b_{i,t+1}^{1} &= \min\left\{B,\ b_{i,t}^{1}+e_{i,t}-\left(c_{\mathrm{rt}}D_{i,t}^{1}+c_{\mathrm{st}}\right)\lambda^{1}\right\}\\
&= \min\left\{B,\ b_{i,t}^{1}+e_{i,t}-\left(c_{\mathrm{rt}}\sum_{p_{j,t}:j\neq i\wedge i,p_{j,t}\in\mathcal{P}_{j,t}}\alpha_{j,t}\cdot\lambda_{j,t}^{1}+c_{\mathrm{st}}\lambda_{i,t}^{1}\right)\right\}\\
&= b_{i,t}^{1}+e_{i,t}-\left(c_{\mathrm{rt}}\sum_{p_{j,t}:j\neq i\wedge i,p_{j,t}\in\mathcal{P}_{j,t}}\alpha_{j,t}\cdot\lambda_{j,t}^{1}+c_{\mathrm{st}}\lambda_{i,t}^{1}\right)=0,
\end{aligned}\tag{8}$$

as $B>0$, where the first and the second line come from all the rates being equal in the first iteration and the fact that all the i's descendants whose path $p_{j,t}$ contains i send $\alpha_{j,t}$ fraction of their flow through i.

As every iteration only increases the rates, if we allow $\lambda_{i,t}$ to be increased in the next iteration, then [from (8)] we would get $b_{i,t+1}<0$, which is a contradiction. Alternatively, if we increase $\lambda_{i,t}^{1}$ at the expense of decreasing some $\lambda_{j,t}^{1}$, $i\in p_{j,t}\backslash\{j\}$, to keep $b_{i,t+1}\geq 0$, then the solution is not max-min fair, as $\lambda_{j,t}^{1}=\lambda_{i,t}^{1}=\lambda^{1}$. This proves the necessity of the rule (F1). By the same observation, if we increase the rate $\lambda_{j,t}^{1}$ of any of the node i's descendants j at time t, we will necessarily get $b_{i,t+}<0$ (or would need to sacrifice the max-min fairness). This proves the rule (F3) for all the descendants of node i, such that $F_{i,t}^{2}$ is set to 0 by the rule (F1).

Now let (i,t), $(i,t-1)$, $(i,t-2),\ldots,(i,\tau)$, $\tau\geq 1$, be the longest sequence with the property that: $b_{i,t}=0$ and $b_{i,s}^{1}+e_{i,s}-\Delta b_{i,s}^{1}\leq B\ \forall s\in\{t,t-1,\ldots,\tau\}$. Observe that when this is the case, we have:

$$\begin{aligned}
&\forall s\in\{\tau,\tau+1,\ldots,t-2,t-1\}:\\
&b_{i,s+1}^{1}=\min\left\{B,b_{i,s}^{1}+e_{i,s}-\Delta b_{i,s}^{1}\right\}=b_{i,s}^{1}+e_{i,s}-\Delta b_{i,s}^{1}\\
&\qquad=b_{i,s}^{1}+e_{i,s}-\left(c_{\mathrm{rt}}\sum_{p_{j,t}:j\neq i\wedge i,p_{j,t}\in\mathcal{P}_{j,t}}\alpha_{j,t}\cdot\lambda_{j,t}^{1}+c_{\mathrm{st}}\lambda_{i,s}^{1}\right)
\end{aligned}$$

This gives a recursive relation, so $b_{i,t+1}$ can also be written as:

$$b^1_{i,t+1} = b^1_{i,\tau} + \sum_{s=\tau}^{t} e_{i,s} - c_{\mathrm{rt}} \sum_{s=\tau}^{t} \sum_{p_{j,t}: j \neq i \wedge i, p_{j,t} \in \mathcal{P}_{j,t}} \alpha_{j,t} \cdot \lambda^1_{j,t} - c_{\mathrm{st}} \sum_{s=\tau}^{t} \lambda^1_{i,s}.$$

If we increase $\lambda_{i,s}$ or $\lambda_{j,s}$, for any j, s such that $j \neq i$ and $i \in \mathcal{P}_{j,s}$, $s \in \{\tau, \tau + 1, \ldots, t-2, t-1\}$, then either $b_{i,t+1}$ becomes negative, or we sacrifice the max-min fairness, as all the rates are equal to λ^1 in the first iteration. This proves rule (F2) and completes the proof for the necessity of rule (F3).

The inductive step suppose that all the rules are necessary for the iterations $1, 2, \ldots k-1$, and consider the iteration k.

Observe that:

(o1) $\lambda_{j,t} \leq \lambda_{i,t}, \forall j : i \in \mathcal{P}_{j,t}$, as all the rates, until they are fixed, get increased by the same amount in each iteration, and once a rate gets fixed for some (i, t), by the rule (F3), it gets fixed for all the node i's descendants in the same time slot. Notice that the inequality is strict only if $\lambda_{j,t}$ got fixed before $\lambda_{i,t}$; otherwise these two rates get fixed to the same value.

(o2) Once fixed, a rate never becomes active again.

(o3) If a rate $\lambda_{i,t}$ gets fixed in iteration k, then $\lambda_{i,t} = \lambda^k_{i,t} = \sum_{p=1}^{k} \lambda^p = \lambda^l_{i,t}, \forall l > k$.

Suppose that $b^k_{i,t+1} = 0$ for some $i \in \{1, .., n\}, t \in \{1, \ldots, T\}$. If $F^k_{i,t} = 0$, then by the inductive hypothesis $\lambda_{i,t}$ cannot be further increased in any of the iterations $k, k+1, \ldots$. Assume $F^k_{i,t} = 1$. Then:

$$\begin{aligned} b^k_{i,t+1} &= \min\left\{B,\ b^k_{i,t} + e_{i,t} - \left(c_{\mathrm{rt}} \sum_{p_{j,t}: j \neq i \wedge i, p_{j,t} \in \mathcal{P}_{j,t}} \alpha_{j,t} \cdot \lambda^k_{j,t} + c_{\mathrm{st}} \lambda^k_{i,t}\right)\right\} \\ &= b^k_{i,t} + e_{i,t} - \left(c_{\mathrm{rt}} \sum_{p_{j,t}: j \neq i \wedge i, p_{j,t} \in \mathcal{P}_{j,t}} \alpha_{j,t} \cdot \lambda^k_{j,t} + c_{\mathrm{st}} \lambda^k_{i,t}\right) = 0. \end{aligned}$$

By the observation (o1), $\lambda^k_{j,t} \leq \lambda^k_{i,t}$, $\forall j$ such that $i \in p_{j,t} \backslash \{j\}$, where the inequality holds with equality if $F^k_{j,t} = 0$. Therefore, if we increase $\lambda_{i,t}$ in some of the future iterations, either $b_{i,t+1} < 0$, or we need to decrease some $\lambda_{j,t} \leq \lambda_{i,t}$, violating the max-min fairness condition. This proves the necessity of the rule (F1). For the rule (F3), as for all (j, t) with $j \neq i$, $F^k_{j,t} = 1$ and $i \in \mathcal{P}_{j,t}$, we have $\lambda_{j,t} = \lambda_{i,t}$, none of the i's descendants can further increase its rate in slot t.

Now for (i, t) such that $b^k_{i,t+1} = 0$, let $(i, t), (i, t-1), (i, t-2), \ldots, (i, \tau), \tau \geq 1$, be the longest sequence with the property that: $b^k_{i,s} + e_{i,s} - \Delta b^k_{i,s} \leq B\ \forall s \in \{t, t-1, \ldots, \tau\}$. Similarly as for the base case:

$$\begin{aligned} \forall s \in &\{\tau, \tau+1, \ldots, t-2, t-1\} : \\ b^k_{i,s+1} &= \min\left\{B, b^k_{i,s} + e_{i,s} - \Delta b^k_{i,s}\right\} \\ &= b^k_{i,s} + e_{i,s} - \left(c_{\mathrm{rt}} \sum_{p_{j,t}: j \neq i \wedge i, p_{j,t} \in \mathcal{P}_{j,t}} \alpha_{j,t} \cdot \lambda^k_{j,t} + c_{\mathrm{st}} \lambda^k_{i,s}\right), \end{aligned}$$

and we get that:

$$b_{i,t+1}^k = b_{i,\tau}^k + \sum_{s=\tau}^{t} e_{i,s} - c_{\mathrm{rt}} \sum_{s=\tau}^{t} \sum_{p_{j,t}: j \neq i \wedge i, p_{j,t} \in \mathcal{P}_{j,t}} \alpha_{j,t} \cdot \lambda_{j,t}^k - c_{\mathrm{st}} \sum_{s=\tau}^{t} \lambda_{i,s}^k. \tag{9}$$

If any of the rates appearing in (9), was fixed in some previous iteration, then it cannot be further increased by the inductive hypothesis. By the observation (o1), all the rates that are active are equal, and all the rates that are fixed are strictly lower than the active rates. Therefore, by increasing any of the active rates from (9), we either violate battery nonnegativity constraint or the max-min fairness criterion. Therefore, rule (F2) holds, and rule (F3) holds for all the descendants of nodes whose rates got fixed by the rule (F2), in the corresponding time slots. □

Lemma 4 (Sufficiency) *If $F_{i,t}^{k+1} = 1$, then $\lambda_{i,t}$ can be further increased by a positive amount in the iteration $k+1$, $\forall i \in \{1, \ldots, n\}$, $\forall t \in \{1, \ldots, T\}$.*

Proof Suppose that $F_{i,t}^{k+1} = 1$. Notice that by increasing $\lambda_{i,t}$ by some $\Delta\lambda_{i,t}$ node i spends an additional $\Delta b_{i,t} = c_{\mathrm{st}} \Delta\lambda_{i,t}$ energy *only in the time slot t*. As $F_{i,t}^{k+1} = 1$, by the rules (F1) and (F2), either $b_{i,t'} > 0\ \forall t' > t$, or there is a time slot $s > t$ such that $b_{i,s}^k + e_{i,s} - \Delta b_{i,s}^k > B$ and $s < s'$, where $s' = \arg\min \{\tau > t : b_{i,\tau} = 0\}$.

If $b_{i,t'} > 0\ \forall t' > t$, then the node i can spend $\Delta b_{i,t} = \min_{t+1 \le t' \le T+1} b_{i,t'}^k$ energy, and keep $b_{i,t'} \ge 0$, $\forall t'$, which follows from the battery evolution (6).

If there is a slot $s' > t$ in which $b_{i,s'}^k = 0$, then let s be the minimum time slot between t and s', such that $b_{i,s}^k + e_{i,s} - \Delta b_{i,s}^k > B$. Decreasing the battery level at s by $(b_{i,s}^k + e_{i,s} - \Delta b_{i,s}^k) - B$ does not influence any other battery levels, as in either case $b_{i,s+1} = B$. As all the battery levels are positive in all the time slots between t and s, i can spend at least $\min\{(b_{i,s}^k + e_{i,s} - \Delta b_{i,s}^k) - B, \quad \min_{t+1 \le t' \le s} b_{i,t'}^k\} > 0$ energy at time t and have $b_{i,t'} \ge 0\ \forall t'$.

By rule (F3), $\forall j$ such that $j \in \mathcal{P}_{i,t}$ we have that $b_{j,t} > 0$, and, furthermore, if $\exists s' > t$ with $b_{j,s'} = 0$ then $\exists s \in \{t, s'\}$ such that $b_{i,s}^k + e_{i,s} - \Delta b_{i,s}^k > B$. By the same observations as for the node i, each $j \in \mathcal{P}_{i,t}$ can spend some extra energy $\Delta b_{j,t} > 0$ in the time slot t and keep all the battery levels nonnegative. In other words, on each directed path $p_{i,t} \in \mathcal{P}_{i,t}$ from the node i to the sink every node can spend some extra energy in time slot t and keep its battery levels nonnegative. Therefore, if we keep all other rates fixed, the rate $\lambda_{i,t}$ can be increased by at least $\Delta\lambda_{i,t} = \min\{\Delta b_{i,t}/c_{\mathrm{st}}, \min_{j \in \mathcal{P}_{i,t}} \Delta b_{j,t}/c_{\mathrm{rt}}\} > 0$.

As each active rate $\lambda_{i,t}$ can (alone) get increased in the iteration $k+1$ by some $\Delta\lambda_{i,t} > 0$, it follows that all the active rates can be increased simultaneously by at least $\min_{i,t} \Delta\lambda_{i,t}/(T(c_{\mathrm{st}} + nc_{\mathrm{rt}})) > 0$. □

Theorem 1 *Fixing rules (F1), (F2) and (F3) provide necessary and sufficient conditions for fixing the rates in* Water-filling-Framework.

Proof Follows directly from Lemmas 3 and 4. □

Lemma 5 Fixing-the-Rates *for* P-Determine-Rates *can be implemented in time* $O(mT)$.

Proof Rules (F1) and (F2) can be implemented for each node independently in time $O(T)$ by examining the battery levels from slot $T+1$ to slot 2.

For the rule (F3), in each time slot $t \in \{1, \dots, T\}$ enqueue all the nodes i whose rates got fixed in time slot t by either of the rules (F1), (F2) and perform a breadth-first search over the graph determined by the enqueued nodes and the edges from $\cup_{j\in\{1,\dots,n\}}\mathcal{P}_{j,t}$ added to the graph with reversed direction. Fix the rates of all the nodes discovered by the breadth-first search. This gives $O(m)$ time per slot, for a total time of $O(mT)$. Combining with the time for rules (F1) and (F2), the result follows. □

Combining Lemmas 2 and 5, we can compute the total running time of Water-filling-Framework for P-Unsplittable-Find, as stated in the following lemma.

Lemma 6 Water-filling-Framework *with Steps 2* Maximizing-the-Rates *and 3* Fixing-the-Rates *implemented as described in Sect. 3 runs in time:*

$$O(nT(mT + nT\log(B + \max_{i,t} e_{i,t}/(\delta c_{st})))).$$

Proof To bound the running time of the overall algorithm that performs lexicographic maximization, we need to first bound the number of iterations that the algorithm performs. As in each iteration at least one sensing rate $\lambda_{i,t}$, $i \in \{1, \dots, n\}$, $t \in \{1, \dots, T\}$, gets fixed, and once fixed remains fixed, the total number of iterations is $O(nT)$. The running time of each iteration is determined by the running times of the Steps 2 (Maximizing-the-Rates) and 3 (Fixing-the-Rates) of the Water-filling-Framework. Maximizing-the-Rates runs in $O\left(nT\log\left(\frac{B+\max_{i,t}e_{i,t}}{\delta c_{\text{st}}}\right)\right)$ (Lemma 2), whereas Fixing-the-Rates runs in $O(mT)$ time (Lemma 5). Therefore, the total running time is: $O\left(nmT^2 + n^2T^2\log\left(B + \max_{i,t} e_{i,t}/(\delta c_{\text{st}})\right)\right)$. □

4 Fractional Routing

Computing a lexicographically maximum fractional routing can be formulated as a generalized flow problem with capacitated nodes, where the nodes' capacities change over time and are determined by the battery states. It is not difficult to see that the feasible region of the rates and flows in P-Fractional-Routing–Routing can be described by the following constraints:

$$\begin{aligned}
&\forall i \in \{1, \dots, n\}, t \in \{1, \dots, T\}:\\
&\quad f_{i,t}^{\Sigma} + \lambda_{i,t} = \sum_{(i,j)\in E} f_{ij,t}\\
&\quad b_{i,t+1} = \min\left\{B,\ b_{i,t} + e_{i,t} - (c_{\text{rt}} f_{i,t}^{\Sigma} + c_{\text{st}}\lambda_{i,t})\right\}\\
&\quad b_{i,t} \geq 0,\ \lambda_{i,t} \geq 0,\ f_{ij,t} \geq 0, \forall (i,j) \in E,
\end{aligned}$$

where $f_{i,t}^{\Sigma} \equiv \sum_{(j,i)\in E} f_{ji,t}$.

We can avoid computing the values of battery levels $b_{i,t+1}$, and instead explicitly write the non-negativity constraints for each of the terms inside the min{.}. This increases the number of constraints from $O(mT)$ to $O(mT^2)$, but will allow us to make more observations about the problem structure. Reordering the terms, we get the following formulation:

$$
\begin{aligned}
&\forall i \in \{1, \ldots, n\}, t \in \{1, \ldots, T\} : \\
&f^{\Sigma}_{i,t} + \lambda_{i,t} = \sum_{(i,j)\in E} f_{ij,t} \qquad (10)\\
&\sum_{\tau=1}^{t} (c_{\text{rt}} f^{\Sigma}_{i,\tau} + c_{\text{st}}\lambda_{i,t}) \leq b_{i,1} + \sum_{\tau=1}^{t} e_{i,\tau} \qquad (11)\\
&\sum_{\tau=s}^{t} (c_{\text{rt}} f^{\Sigma}_{i,\tau} + c_{\text{st}}\lambda_{i,t}) \leq B + \sum_{\tau=s}^{t} e_{i,\tau}, \quad 2 \leq s \leq t \qquad (12)\\
&\lambda_{i,t} \geq 0,\ f_{ij,t} \geq 0, \forall (i, j) \in E \qquad (13)
\end{aligned}
$$

In the kth iteration of WATER-FILLING-FRAMEWORK we have that $\lambda^k_{i,t} = \lambda^{k-1}_{i,t} + F^k_{i,t} \cdot \lambda^k = \sum_{l=1}^{k} F^l_{i,t} \cdot \lambda^l$, where $\lambda^0_{i,t} = 0$. Let:

$$
u^b_{i,t} = b_{i,1} + \sum_{\tau=1}^{t} \left(e_{i,\tau} - c_{\text{st}}\lambda^{k-1}_{i,\tau}\right),
$$

$$
u^B_{i,t,s} = B + \sum_{\tau=s}^{t} \left(e_{i,\tau} - c_{\text{st}}\lambda^{k-1}_{i,\tau}\right).
$$

Since in the iteration k all $\lambda^{k-1}_{i,t}$'s are constants, the rate maximization subproblem can be written as:

$$
\begin{aligned}
\textbf{max} \quad & \lambda^{\mathbf{k}} \\
\textbf{s.t.} \quad & \forall i \in \{1, \ldots, n\}, t \in \{1, \ldots, T\} : \qquad (14)\\
& -f^{\Sigma}_{i,t} - F^k_{i,t} \cdot \lambda^k + \sum_{(i,j)\in E} f_{ij,t} = \lambda^{k-1}_{i,t} \qquad (15)\\
& \sum_{\tau=1}^{t} (c_{\text{rt}} f^{\Sigma}_{i,\tau} + F^k_{i,\tau} \cdot c_{\text{st}}\lambda^k) \leq u^b_{i,t} \qquad (16)\\
& \sum_{\tau=s}^{t} (c_{\text{rt}} f^{\Sigma}_{i,\tau} + F^k_{i,\tau} \cdot c_{\text{st}}\lambda^k) \leq u^B_{i,t,s}, \quad 2 \leq s \leq t \qquad (17)\\
& \lambda^k \geq 0,\ f_{ij,t} \geq 0, \forall (i, j) \in E \qquad (18)
\end{aligned}
$$

Notice that in this formulation all the variables are on the left-hand side of the constraints, whereas all the right-hand sides are constant.

4.1 Relation to Multi-commodity Flow

Let $T = 2$, and consider the constraints in (10)–(13). We claim that verifying whether any set of sensing rates $\lambda_{i,t}$ is feasible is at least as hard as solving a 2-commodity feasible flow problem with capacitated nodes and a single sink:

Claim Any 2-commodity feasible flow problem with capacitated nodes and a single sink can be reduced to a feasible flow problem in an energy harvesting network over a time horizon $T = 2$.

Proof To prove the claim, we first rewrite the constraints in (10)–(13) as:

$$\sum_{(j,i)\in E} f_{ji,t} + \lambda_{i,t} = \sum_{(i,j)\in E} f_{ij,t}, \quad t \in \{1,2\}$$

$$\sum_{(j,i)\in E} f_{ji,1} \le \frac{1}{c_{\text{rt}}}(b_{i,1} + e_{i,1} - c_{\text{st}}\lambda_{i,1})$$

$$\sum_{\tau=1}^{2}\sum_{(j,i)\in E} f_{ji,\tau} \le \frac{1}{c_{\text{rt}}}\Big(b_{i,1} + \sum_{\tau=1}^{2}\big(e_{i,\tau} - c_{\text{st}}\lambda_{i,\tau}\big)\Big)$$

$$\sum_{(j,i)\in E} f_{ji,2} \le \frac{1}{c_{\text{rt}}}(B + e_{i,2} - c_{\text{st}}\lambda_{i,2})$$

$$\lambda_{i,t} \ge 0,\ f_{ij,t} \ge 0,\ \ \forall i \in \{1,\ldots,n\}, (i,j) \in E, t \in \{1,2\}$$

Suppose that we are given any 2-commodity flow problem with capacitated nodes, and let:

– $\lambda_{i,t}$ denote the supply of commodity t at node i;
– $u_{i,t}$ denote the per-commodity capacity constraint at node i for commodity t;
– u_i denote the bundle capacity constraint at node i.

Choose values of $c_{\text{s}}, c_{\text{rt}}, B, b_{i,1}, b_{i,2}, e_{i,1}, e_{i,2}$ so that the following equalities are satisfied:

$$u_{i,1} = \frac{1}{c_{\text{rt}}}\cdot\big(b_{i,1} + e_{i,1} - c_{\text{st}}\lambda_{i,1}\big)$$

$$u_{i,2} = \frac{1}{c_{\text{rt}}}\cdot\big(B + e_{i,2} - c_{\text{st}}\lambda_{i,2}\big)$$

$$u_i = \frac{1}{c_{\text{rt}}}(b_{i,1} + \sum_{\tau=1}^{2}\big(e_{i,\tau} - c_{\text{st}}\lambda_{i,\tau}))$$

Then feasibility of the given 2-commodity flow problem is equivalent to the feasibility of (10)–(13). Therefore, any 2-commodity feasible flow problem can be stated as an equivalent problem of verifying feasibility of sensing rates $\lambda_{i,t}$ in an energy harvesting network for $T = 2$. □

For $T > 2$, (11) and (12) are general packing constraints. If a flow graph G_t in time slot t is observed as a flow of a commodity indexed by t, then for each node i the constraints (11) and (12) define capacity constraints for every sequence of consecutive commodities $s, s+1, \dots, t$, $1 \le s \le t \le T$.

Therefore, to our current knowledge, it is unlikely that the general rate assignment problem can be solved exactly in polynomial time without the use of linear programming, as there have not been any combinatorial algorithms that solve feasible 2-commodity flow optimally.

4.2 Fractional Packing Approach

The fractional packing problem is defined as follows [30]:

PACKING: *Given a convex set P for which $Ax \ge 0\ \forall x \in P$, is there a vector x such that $Ax \le b$?* Here, A is a $p \times q$ matrix, and x is a q-length vector.

A vector x is an ϵ-approximate solution to the PACKING problem if $x \in P$ and $Ax \le (1+\epsilon)b$. Alternatively, scaling all the constraints by $\frac{1}{1+\epsilon}$, we obtain a solution $x' = \frac{1}{1+\epsilon}x \in (\frac{1}{1+\epsilon}x_{\mathrm{OPT}}, x_{\mathrm{OPT}}] \subset ((1-\epsilon)x_{\mathrm{OPT}}, x_{\mathrm{OPT}}]$, for $\epsilon < 1$, where x_{OPT} is an optimal solution to the packing problem. The algorithm in [30] either provides an ϵ-approximate solution to the PACKING problem, or it proves that no such solution exists. Its running time depends on:

- The running time required to solve $\min\{cx : x \in P\}$, where $c = y^T A$, y is a given p-length vector, and $(.)^T$ denotes the transpose of a vector.
- The width of P relative to $Ax \le b$, which is defined by $\rho = \max_i \max_{x \in P} \frac{a_i x}{b_i}$, where a_i is the ith row of A, and b_i is the ith element of b.

For a given error parameter $\epsilon > 0$, a feasible solution to the problem $\min\{\beta : Ax \le \beta b, x \in P\}$, its dual solution y, and $C_{\mathcal{P}}(y) = \min\{cx : c = y^T A, x \in P\}$, [30] defines the following relaxed optimality conditions:

$$(1-\epsilon)\beta y^T b \le y^T Ax \quad (\mathcal{P}1)$$
$$y^T Ax - C_{\mathcal{P}}(y) \le \epsilon(y^T Ax + \beta y^T b) \quad (\mathcal{P}2)$$

The packing algorithm [30] is implemented through subsequent calls to the procedure IMPROVE-PACKING:

Algorithm 2 IMPROVE-PACKING(x, ϵ)[30]

```
1: Initialize $\beta_0 = \max_i a_i x/b_i$; $\alpha = 4\beta_0^{-1}\epsilon^{-1}\ln(2p\epsilon^{-1})$; $\sigma = \epsilon/(4\alpha\rho)$.
2: while $\max_i a_i x/b_i \ge \beta_0/2$ and $x, y$ do not satisfy $(\mathcal{P}2)$ do
3:     For each $i = 1, 2, \dots, p$: set $y_i = (1/b_i)e^{\alpha a_i x/b_i}$.
4:     Find a min-cost point $\widehat{x} \in P$ for costs $c = y^T A$.
5:     Update $x = (1-\sigma)x + \sigma\widehat{x}$.
6: return $x$.
```

The running time of the ϵ-approximation algorithm provided in [30], for $\epsilon \in (0, 1]$, equals $O(\epsilon^{-2}\rho \log(m\epsilon^{-1}))$ multiplied by the time needed to solve $\min\{cx : c = y^T A, x \in P\}$ and compute Ax (Theorem 2.5 in [30]).

4.2.1 Maximizing the Rates as Fractional Packing

We discussed at the beginning of this section that for the kth iteration Maximize-the-Rates can be stated as (14)–(18). Observe the constraints (16) and (17). Since λ^k, $f_{ij,t}$ and all the right-hand sides in (16) and (17) are nonnegative, (16) and (17) imply the following inequalities:

$$\begin{aligned}
&\forall i \in \{1, \dots, n\}, t \in \{1, \dots, T\}: \\
&\quad F^k_{i,\theta} \cdot c_{\text{st}}\lambda^k \le u^b_{i,t}, \qquad 1 \le \theta \le t \\
&\quad F^k_{i,\theta} \cdot c_{\text{st}}\lambda^k \le u^B_{i,t,s}, \qquad 2 \le s \le t,\ s \le \theta \le t \\
&\quad c_{\text{rt}} \sum_{(j,i)\in E} f_{ji,\theta} \le u^b_{i,t} - c_{\text{st}} \sum_{\tau=1}^{t} F^k_{i,\tau}\lambda^k, \qquad 1 \le \theta \le t \\
&\quad c_{\text{rt}} \sum_{(j,i)\in E} f_{ji,\theta} \le u^B_{i,t,s} - c_{\text{st}} \sum_{\tau=s}^{t} F^k_{i,\tau}\lambda^k, \qquad 2 \le s \le t,\ s \le \theta \le t
\end{aligned}$$

Therefore, we can yield an upper bound $\lambda^k_{\max}$ for λ^k:

$$\lambda^k \le \lambda^k_{\max} \equiv \frac{1}{c_{\text{st}}} \min_{i,t,s\ge 2} \left\{ u^b_{i,t} : \sum_{\tau=1}^{t} F^k_{i,\tau} > 0,\ u^B_{i,t,s} : \sum_{\tau=s}^{t} F^k_{i,\tau} > 0 \right\} \tag{19}$$

For a fixed λ^k, the flow entering a node i at time slot t can be bounded as:

$$\sum_{(j,i)\in E} f_{ji,t} \le u_{i,t} \equiv \frac{1}{c_{\text{rt}}} \min_{\substack{i,t_1\ge t \\ s\ge 2}} \left\{ u^b_{i,t_1} - c_{\text{st}} \sum_{\tau=1}^{t_1} F^k_{i,\tau}\lambda^k,\ u^B_{i,t,s} - c_{\text{st}} \sum_{\tau=s}^{t_1} F^k_{i,\tau}\lambda^k \right\} \tag{20}$$

We choose to keep only the flows $f_{ij,t}$ as variables in the Packing problem. Given a $\lambda^k \in [0, \lambda^k_{\max}]$, we define the convex set P[6] via the following set of constraints:

$$\forall i \in \{1, \dots, n\}, t \in \{1, \dots, T\}:$$

$$-\sum_{(j,i)\in E} f_{ji,t} + \sum_{(i,j)\in E} f_{ij,t} = \lambda^{k-1}_{i,t} + F^k_{i,t} \cdot \lambda^k \tag{21}$$

$$\sum_{(j,i)\in E} f_{ji,t} \le u_{i,t} \tag{22}$$

$$f_{ij,t} \ge 0, \quad \forall (i,j) \in E \tag{23}$$

[6] P is determined by linear equalities and inequalities, which implies that it is convex.

Proposition 1 *For P described by (21)–(23) and a given vector y, problem* $\min\{cf : c = y^T Af,\ f \in P\}$ *can be solved via T min-cost flow problems.*

Proof Constraint (21) is a standard flow balance constraint at a node i in a time slot t, whereas constraint (22) corresponds to a node capacity constraint at the time t, given by (20). As there is no interdependence of flows over time slots, we get that the problem can be decomposed into subproblems corresponding to individual time slots. Therefore, to solve the problem $\min\{cf : c = y^T Af,\ f \in P\}$ for a given vector y, it suffices to solve T min-cost flow problems, one for each time slot $t \in \{1, 2, \ldots, T\}$. □

The remaining packing constraints of the form $Ax \leq b$ are given by (16) and (17), where $x \equiv f$.

Proposition 2 $Ax \geq 0\ \forall f \in P$.

Proof As $f_{ij,t} \geq 0\ \forall (i, j) \in E, t \in \{1, \ldots, T\}$, and all the coefficients multiplying $f_{ij,t}$'s in (16) and (17) are nonnegative, the result follows immediately. □

Lemma 7 *One iteration of* Improve-Packing *for* P-Fractional-Routing *can be implemented in time*

$$O\left(nT^2 + T \cdot MCF(n, m)\right),$$

where $MCF(n, m)$ denotes the running time of a min-cost flow algorithm on a graph with n nodes and m edges.

Proof Since the flows over edges appear in the packing constraints only as the sum-terms of the total incoming flow of a node i in a time slot t, we can use the total incoming flow $f^{\Sigma}_{i,t} = \sum_{(j,i)\in E} f_{ji,t}$ for each (i, t) as variables. Reordering the terms, the packing constraints can be stated as:

$$\sum_{\tau=1}^{t} f^{\Sigma}_{i,\tau} \leq \frac{1}{c_{\mathrm{rt}}}\left(u^{b}_{i,t} - c_{\mathrm{st}} \sum_{\tau=1}^{t} F^{k}_{i,\tau}\lambda^{k}\right), \quad 1 \leq t \leq T \tag{24}$$

$$\sum_{\tau=s}^{t} f^{\Sigma}_{i,\tau} \leq \frac{1}{c_{\mathrm{rt}}}\left(u^{B}_{i,t,s} - c_{\mathrm{st}} \sum_{\tau=s}^{t} F^{k}_{i,\tau}\lambda^{k}\right), \ 2 \leq s \leq t,\ 2 \leq t \leq T \tag{25}$$

With this formulation on hand, the matrix A of the packing constraints $Af^{\Sigma} \leq b$ is a 0–1 matrix that can be decomposed into blocks of triangular matrices. To see this, first notice that for each node i constraints given by (24) correspond to a lower-triangular 0–1 matrix of size T. Each sequence of constraints of type (25) for fixed i and fixed $s \in \{2, \ldots, T\}$, and $t \in \{s, s+1, \ldots, T\}$ corresponds to a lower-triangular 0–1 matrix of size $T - s + 1$. This special structure of the packing constraints matrix allows an efficient computation of the dual vector y and the corresponding cost vector c. Moreover, as constraints (24, 25) can be decomposed into independent blocks of constraints of the type $A_i f^{\Sigma}_i \leq b_i$ for nodes $i \in \{1, \ldots, n\}$, the dual vector y and the corresponding cost vector c can be decomposed into vectors y_i, c_i for $i \in \{1, \ldots, n\}$. Cost $c_{i,t}$ can be interpreted as the cost of sending 1 unit of flow through node i in time slot t.

Observe the block of constraints $A_i f_i^{\Sigma} \leq b_i$ corresponding to the node i. The structure of A_i is as follows:

$$
\begin{array}{r}
T \left\{ \begin{matrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{matrix} \right. \\
T-1 \left\{ \begin{matrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & 1 & \cdots & 1 & 1 \end{matrix} \right. \\
\vdots \qquad \\
2 \left\{ \begin{matrix} 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 1 \end{matrix} \right. \\
1 \left\{ \begin{matrix} 0 & 0 & 0 & \cdots & 0 & 1 \end{matrix} \right.
\end{array}
$$

As A_i can be decomposed into blocks of triangular matrices, each $y_{i,j}$ in the IMPROVE-PACKING procedure can be computed in constant time, yielding $O\left(\frac{T(T-1)}{2}\right) = O\left(T^2\right)$ time for computing y_i. This special structure of A_i also allows a fast computation of the cost vector c_i. Observe that each $c_{i,t}, t \in \{1, \ldots, T\}$ can be computed by summing $O(T)$ terms. For example, $c_{i,1} = \sum_{j=1}^{T} y_{i,j}$, $c_{i,2} = c_{i,1} - y_{i,1} + \sum_{j=T+1}^{2T-1} y_{i,j}$, $c_{i,3} = c_{i,2} - y_{i,2} - y_{i,T+1} + \sum_{j=2T}^{3T-2} y_{i,j}$, etc. Therefore, computing the costs for node i takes $O(T^2)$ time. This further implies that one iteration of IMPROVE-PACKING takes $O\left(nT^2 + T \cdot MCF(n,m)\right)$ time, where $MCF(n,m)$ denotes the running time of a min-cost flow algorithm on a graph with n nodes and m edges. □

Lemma 8 *Width ρ of P relative to the packing constraints* (16) *and* (17) *is $O(T)$.*

Proof As $u_{i,t}$ is determined by the tightest constraint in which $\sum_{(j,i)\in E} f_{ji,t} \equiv f_{i,t}^{\Sigma}$ appears, we have that in every constraint given by (24) and (25):

$$f_{i,\theta}^{\Sigma} \leq \frac{1}{c_{\text{rt}}}(u_{i,t}^{b} - c_{\text{st}} \sum_{\tau=1}^{t} F_{i,\tau}^{k} \lambda^{k}), \quad 1 \leq \theta \leq t$$

$$f_{i,\theta}^{\Sigma} \leq \frac{1}{c_{\text{rt}}}(u_{i,t,s}^{B} - c_{\text{st}} \sum_{\tau=s}^{t} F_{i,\tau}^{k} \lambda^{k}), \ 2 \leq s \leq t, s \leq \theta \leq t$$

As the sum of $f_{ij,\theta}^{\Sigma}$ over θ in any constraint from (24, 25) can include at most T terms, it follows that $\rho \leq \frac{T \cdot b_i}{b_i} = T$. □

Lemma 9 MAXIMIZING-THE-RATES *that uses packing algorithm from* [30] *can be implemented in time: $\widetilde{O}(T^2\epsilon^{-2} \cdot (nT + MCF(n,m)))$, where $\widetilde{O}$-notation ignores poly-log terms.*

Proof We have from (19) that $\lambda^k \in [0, \lambda^k_{\max}]$, therefore, we can perform a binary search to find the maximum λ^k for which both $\min\{y^T Af | f \in P\}$ is feasible and Packing outputs an ϵ-approximate solution. Multiplying the running time of the binary search by the running time of the packing algorithm [30], the total running time becomes:

$$O\left(\log\left(\frac{\lambda^k_{\max}}{\delta}\right)\epsilon^{-2}\rho\log(m\epsilon^{-1})\left(nT^2 + T\cdot MCF(n,m)\right)\right)$$
$$= \widetilde{O}\left(\frac{T^2}{\epsilon^2}\cdot(nT + MCF(n,m))\right).$$

□

4.2.2 Fixing the Rates

As Maximizing-the-Rates described in previous subsection outputs an ϵ-approximate solution in each iteration, the objective of the algorithm is not to output a max-min fair solution anymore, but an ϵ-approximation. We consider the following notion of approximation, as in [21]:

Definition 3 For a problem of lexicographic maximization, say that a feasible solution given as a vector v is an element-wise ϵ-approximate solution, if for vectors v and v_{OPT} sorted in nondecreasing order $v \geq (1-\epsilon)v_{\text{OPT}}$ component-wise, where v_{OPT} is an optimal solution to the given lexicographic maximization problem.

Let Δ be the smallest real number that can be represented in a computer, and consider the algorithm that implements Fixing-the-Rates as stated below.

Algorithm 3 Fixing-the-Rates

```
1: Solve the following linear program:
2: max $\sum_{i=1}^{n} F^k_{i,t}\lambda^k_{i,t}$
3:   s.t. $\forall i \in \{1,\ldots,n\}, t \in \{1,\ldots,T\}$ :
4:       $\lambda^k_{i,t} \geq \lambda^{k-1}_{i,t} + F^k_{i,t}\cdot\lambda^k$
5:       $\lambda^k_{i,t} \leq \lambda^{k-1}_{i,t} + F^k_{i,t}\cdot\left(\epsilon\lambda^{k-1}_{i,t} + (1+\epsilon)\lambda^k + \Delta\right)$
6:       $f^{\Sigma}_{i,t} + \lambda^k_{i,t} = \sum_{(i,j)\in E} f_{ij,t}$
7:       $b_{i,t+1} = \min\left\{B,\ b_{i,t} + e_{i,t} - \left(c_{\text{rt}} f^{\Sigma}_{i,t} + c_{\text{st}}\lambda^k_{i,t}\right)\right\}$
8:       $b_{i,t} \geq 0, \quad \lambda^k_{i,t} \geq 0, \quad f_{ij,t} \geq 0$
9: Let $F^{k+1}_{i,t} = F^k_{i,t}, \forall i, t.$
10: If $\lambda^k_{i,t} < (1+\epsilon)(\lambda^{k-1}_{i,t} + F^k_{i,t}\cdot\lambda^k) + \Delta$, set $F^{k+1}_{i,t} = 0$.
```

Assume that Fixing-the-Rates does not change any of the rates, but only determines what rates should be fixed in the next iteration, i.e., it only makes (global) changes to $F^{k+1}_{i,t}$. Then:

Lemma 10 *If the* Steps 2 *and* 3 *in the* Water-filling-Framework *are implemented as* Maximizing-the-Rates *and* Fixing-the-Rates *from this section, then* *the solution output by the algorithm is an element-wise ϵ-approximate solution to the lexicographic maximization of* $\lambda_{i,t} \in \mathcal{R}$.

Proof The proof is by induction.

The base case observe the first iteration of the algorithm. After rate maximization, $\forall i, t : \lambda_{i,t} = \lambda^1 \geq \frac{1}{1+\epsilon}\lambda^1_{\text{OPT}}$ and $F^1_{i,t} = 1$.

Observe that in the output of the linear program of FIXING-THE-RATES, all the rates must belong to the interval $[\lambda^1, (1+\epsilon)\lambda^1 + \Delta]$. Choose any (i,t) with $\lambda^1_{i,t} < (1+\epsilon)(\lambda^{k-1}_{i,t} + F^1_{i,t} \cdot \lambda^1) + \Delta = (1+\epsilon)\lambda^1 + \Delta$. There must be at least one such rate, otherwise the rate maximization did not return an ϵ-approximate solution. As $\sum_{i=1}^n F^1_{i,t}\lambda^1_{i,t} = \sum_{i=1}^n \lambda^1_{i,t}$ is maximum, if $\lambda^1_{i,t}$ is increased, then at least one other rate needs to be decreased to maintain feasibility. To get a lexicographically greater solution $\lambda^1_{i,t}$ can only be increased by lowering the rates with the value greater than $\lambda^1_{i,t}$. Denote by $S^1_{i,t}$ the set of all the rates $\lambda^1_{j,\tau}$ such that $\lambda^1_{j,\tau} > \lambda^1_{i,t}$. In the lexicographically maximum solution, the highest value to which $\lambda^1_{i,t}$ can be increased is at most $\frac{1}{|S^1_{i,t}|}\left(\lambda^1_{i,t} + \sum_{\lambda_{j,\tau} \in S^1_{i,t}} \lambda^1_{j,\tau}\right) < (1+\epsilon)\lambda^1 + \Delta$, which implies $\lambda_{i,t,\max} \leq (1+\epsilon)\lambda^1$. Therefore, if $\lambda_{i,t}$ is fixed to the value of λ^1, it is guaranteed to be in the ϵ-range of its optimal value.

Now consider all the (i,t)'s with $\lambda^1_{i,t} = (1+\epsilon)\lambda^1 + \Delta$. As all the rates that get fixed are fixed to a value $\lambda_{i,t} = \lambda^1 \leq \lambda^1_{i,t}$, it follows that in the next iteration all the rates that did not get fixed can be increased by at least $\epsilon\lambda^1 + \Delta$, which FIXING-THE-RATES properly determines.

The inductive step suppose that up to iteration $k \geq 2$ all the rates that get fixed are in the ϵ-optimal range, and observe the iteration k. All the rates that got fixed prior to iteration k satisfy:

$$\lambda^k_{i,t} \geq \lambda^{k-1}_{i,t} + F^k_{i,t} \cdot \lambda^k = \lambda^{k-1}_{i,t} \text{ , and}$$
$$\lambda^k_{i,t} \leq \lambda^{k-1}_{i,t} + F^k_{i,t} \cdot \left(\epsilon\lambda^{k-1}_{i,t} + (1+\epsilon)\lambda^k + \Delta\right) = \lambda^{k-1}_{i,t}$$

and, therefore, they remain fixed for the next iteration, as $\lambda^k_{i,t} = \lambda^{k-1}_{i,t} < (1+\epsilon)\lambda^{k-1}_{i,t}$.

Now consider all the (i,t)'s with $F^k_{i,t} = 1$. We have that:

$$\lambda^k_{i,t} \geq \lambda^{k-1} + 1 \cdot \lambda^k = \sum_{l=1}^{k} \lambda^l$$

$$\lambda^k_{i,t} \leq (1+\epsilon)\left(\lambda^{k-1} + 1 \cdot \lambda^k\right) + \Delta = (1+\epsilon)\sum_{l=1}^{k} \lambda^l + \Delta$$

Similarly as in the base case, if $\lambda^k_{i,t} < (1+\epsilon)\sum_{l=1}^k \lambda^l + \Delta$, let $S^k_{i,t} = \{\lambda^k_{j,\tau} : \lambda^k_{j,\tau} > \lambda^k_{i,t}\}$. There must be at least one such (i,t), otherwise the rate maximization did not output an ϵ-approximate solution. In any lexicographically greater solution:

$$\lambda_{i,t,\max}^k \le \frac{1}{|S_{i,t}^k|}\left(\lambda_{i,t}^k + \sum_{\lambda_{j,\tau}^k \in S_{i,t}^k} \lambda_{j,\tau}\right)$$
$$< (1+\epsilon)\sum_{l=1}^{k}\lambda^l + \Delta,$$

which implies $\lambda_{i,t,\max}^k \le (1+\epsilon)\sum_{l=1}^k \lambda^l$. Therefore, if we fix $\lambda_{i,t}$ to the value $\sum_{l=1}^k \lambda^l$, it is guaranteed to be at least as high as $(1-\epsilon)$ times the value it gets in the lexicographically maximum solution.

Finally, all the (i, t)'s with $\lambda_{i,t}^k = (1+\epsilon)\sum_{l=1}^k \lambda^l + \Delta$ can simultaneously increase their rates by at least $\epsilon\sum_{l=1}^k \lambda^l + \Delta$ in the next iteration, so it should be $F_{i,t}^{k+1} = 1$, which agrees with FIXING-THE-RATES. □

Lemma 11 *An FPTAS for* P-FRACTIONAL-ROUTING *can be implemented in time:*

$$\widetilde{O}(nT(T^2\epsilon^{-2}\cdot(nT + MCF(n,m) + LP(mT,nT))),$$

where $LP(mT, nT)$ denotes the running time of a linear program with mT variables and nT constraints, and $MCF(n, m)$ denotes the running time of a min-cost flow algorithm run on a graph with n nodes and m edges.

Proof It was demonstrated in the proof of Lemma 10 that in every iteration at least one rate gets fixed. Therefore, there can be at most $O(nT)$ iterations. From Lemma 9, MAXIMIZING-THE-RATES can be implemented in time $\widetilde{O}(T^2\epsilon^{-2}\cdot(nT + MCF(n,m)))$. The time required for running FIXING-THE-RATES is $LP(mT, nT)$, where $LP(mT, nT)$ denotes the running time of a linear program with mT variables and nT constraints. □

Note 1 A linear programming framework as in [8,24,31] when applied to P-FRACTIONAL-ROUTING would yield a running time equal to $O(n^2T^2\cdot LP(mT,nT))$. As the running time of an iteration in our approach is dominated by $LP(mT, nT)$, the improvement in running time is at least $O(nT)$-fold, at the expense of providing an ϵ-approximation.

5 Fixed Fractional Routing

Suppose that we want to solve lexicographic maximization of the rates keeping both the routing and the rates constant over time. Observe that, as both the routing and the rates do not change over time, the energy consumption per time slot of each node i is also constant over time and equal to $\Delta b_i = c_{\mathrm{st}}\lambda_i + c_{\mathrm{rt}}\sum_{(j,i)\in E} f_{ji}$.

Proposition 3 *Maximum constant energy consumption Δb_i can be determined in time $O(T\log(\frac{b_{i,1}+e_{i,1}}{\delta}))$ for each node $i \in V\backslash\{s\}$, for the total time of $O(nT\log(\frac{b_{i,1}+e_{i,1}}{\delta}))$.*

Proof Since the battery evolution can be stated as:

$$b_{i,t+1} = \min\left\{B, \quad b_{i,t} + e_{i,t} - \Delta b_i\right\},$$

maximum Δb_i for which $b_{i,t+1} \geq 0\ \forall t \in \{1, \ldots, T\}$ can be determined via a binary search from the interval $[0, b_{i,1} + e_{i,1}]$, for each node i. □

Similarly as in previous sections, let $F_i^k = 0$ if the rate i is fixed at the beginning of iteration k, and $F_i^k = 1$ if it is not. Initially: $F_i^1 = 1, \forall i$. Rate maximization can then be implemented as follows:

Algorithm 4 MAXIMIZING-THE-RATES(G, F^k, b, e, k)

```
1: $\lambda^k_{\max} = \frac{1}{c_{\mathrm{st}}} \min_i\{\Delta b_i - c_{\mathrm{st}}\lambda_i^{k-1} : F_i^k = 1\}$
2: repeat for $\lambda^k \in [0, \lambda^k_{\max}]$, via binary search
3:     Set the supply of node $i$ to $d_i = \lambda^{k-1} + F_i^k\lambda^k$, capacity of node $i$ to $u_i = \frac{1}{c_{\mathrm{rt}}}(\Delta b_i - c_{\mathrm{st}}\lambda^k)$, for
           each $i$
4:     Set the demand of the sink to $\sum_i d_i$
5:     Solve feasible flow problem on $G$
6: until $\lambda^k$ takes maximum value for which the flow problem is feasible on $G$
```

The remaining part of the algorithm is to determine which rates should be fixed at the end of iteration k. We note that in each iteration k, the maximization of the rates produces a flow f in the graph G^k with the supply rates λ_i^k. Instead of having capacitated nodes, we can modify the input graph by a standard procedure of splitting each node i into two nodes i' and i'', and assigning the capacity of i to the edge (i', i''). This allows us to obtain a residual graph $G^{r,k}$ for the given flow. We claim the following:

Lemma 12 *The rate λ_i of a node $i \in G$ can be further increased in the iteration $k+1$ if and only if there is a directed path from i to the sink in $G^{r,k}$.*

Proof First, observe that the only capacitated edges in G^k are those corresponding to the nodes that were split. The capacity of an edge (i', i'') corresponds to the maximum per-slot energy the node i can spend without violating the battery non-negativity constraint. If an edge (i', i'') has residual capacity of $u^r_{(i',i'')} > 0$, then the node i can spend additional $c_{\mathrm{rt}}u^r_{(i',i'')}$ amount of energy keeping the battery level non-negative in all the time slots. If (i', i'') has no residual capacity ($u^r_{(i',i'')} = 0$), then the battery level of node i reaches zero in at least one time slot, and increasing the energy consumption per time slot leads to $b_{i,t} < 0$ for some t, which is infeasible.

($\Leftarrow$) Suppose that the residual graph contains no directed path from the node i to the sink. By the flow augmentation theorem [1], the flow from the node i cannot be increased even when the flows from all the remaining nodes are kept constant. As the capacities correspond to the battery levels at the nodes, sending more flow from i causes at least one node's battery level to become negative.

($\Rightarrow$) Suppose that there is a directed path from i to the sink, and let $u_i^r > 0$ denote the minimum residual capacity of the edges (split nodes) on that path. Then each node on the path can spend at least $c_{\mathrm{rt}}u_i^r$ amount of energy maintaining feasibility. Let U denote the set of all the nodes that have a directed path to the sink in $G^{r,k}$. Then increasing the rate of each node $i \in U$ by $\Delta\lambda = \dfrac{\min_i u_i^r c_{\mathrm{rt}}}{c_{\mathrm{st}} + nc_{\mathrm{rt}}} > 0$ and augmenting the flows of $i \in U$ over their augmenting paths in $G^{r,k}$ each node on any augmenting path spends at most $\min_i u_i^r c_{\mathrm{rt}}$ amount of energy, which is at most equal to the energy the node is allowed to spend maintaining feasibility. □

Lemma 13 Water-filling-Framework *for* P-Fixed-Fractional-Routing *can be implemented in time*

$$O\Big(n \log\Big(\max_i \Big(\frac{b_{i,1} + e_{i,1}}{\delta}\Big)\Big)(T + MF(n,m))\Big),$$

where $MF(n,m)$ denotes the running time of a max-flow algorithm for a graph with n nodes and m edges.

Proof From Proposition 3, determining the values of Δb_i for $i \in V\backslash\{s\}$ can be implemented in time $O(nT\log(\frac{b_{i,1}+e_{i,1}}{\delta}))$.

Running time of an iteration of Water-filling-Framework is determined by the running times of Maximizing-the-Rates and Fixing-the-Rates. Each iteration of the binary search in Maximizing-the-Rates constructs and solves a feasible flow problem, which is dominated by the time required for running a max-flow algorithm that solves feasible flow problem on the graph G. Therefore, Maximizing-the-Rates can be implemented in time $O(\log(\frac{b_{i,1}+e_{i,1}}{\delta})MF(n,m))$, where $MF(n,m)$ denotes the running time of a max-flow algorithm.

Fixing-the-Rates constructs a residual graph $G^{r,k}$ and runs a breadth-first search on this graph, which can be implemented in time $O(n+m)$ $(= O(MF(n,m))$ for all the existing max-flow algorithms).

Every iteration of Water-filling-Framework fixes at least one of the rates λ_i, $i \in V\backslash\{s\}$, which implies that there can be at most n iterations.

Therefore, the total running time is

$$O\Big(n \log\Big(\max_i \Big(\frac{b_{i,1} + e_{i,1}}{\delta}\Big)\Big)(T + MF(n,m))\Big).$$

□

6 Determining a Routing

In this section we demonstrate that solving P-Unsplittable-Routing and P-Routing-Tree is NP-hard for both problems. Moreover, we show that it is NP-hard to obtain an approximation ratio better than $\Omega(\log n)$ for P-Routing-Tree. For P-Unsplittable-Routing, we design an efficient combinatorial algorithm for a relaxed version of this problem–it determines a time-invariable unsplittable routing that maximizes the minimum rate.

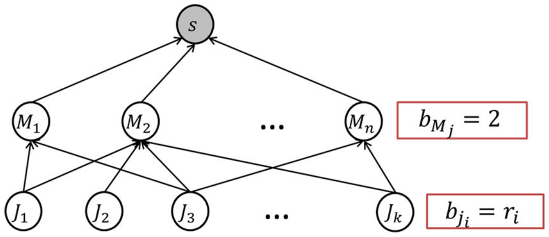


Fig. 7 A reduction from P-NON-UNIFORM-LOAD-BALANCING for proving NP-hardness of P-UNSPLITTABLE-ROUTING. Jobs are represented by nodes J_i, machines by nodes M_j, and there is an edge from J_i to M_j if job J_i can be executed on machine M_j. Each job J_i has time requirement $r_i \in \{1/2, 1\}$, and $\sum_{i=1}^{k} J_i = n$. Available energies at the nodes are shown in the *boxes* next to the nodes. If at the optimum of P-UNSPLITTABLE-ROUTING $\lambda_{J_i} = r_i$ and $\lambda_{M_j} = 1$, then there is an assignment of jobs to the machines such that the sum requirement of jobs assigned to each machine equals 1

6.1 Unsplittable Routing

Lemma 14 P-UNSPLITTABLE-ROUTING *is NP-hard.*

Proof The proof of NP-hardness for P-UNSPLITTABLE-ROUTING is a simple extension of the proof of NP-hardness for max-min fair unsplittable routing provided in[21]. We use the same reduction as in [21], derived from the non-uniform load balancing problem [22]. From [21,22], the following problem is NP-hard:

P-NON-UNIFORM-LOAD-BALANCING: Let $J = \{J_1, \ldots, J_k\}$ be a set of jobs, and $M = \{M_1, \ldots, M_n\}$ be a set of machines. Each job J_i has a time requirement $r_i \in \{1/2, 1\}$, and the sum of all the job requirements is equal to n: $\sum_{i=1}^{k} r_i = n$. Each job $J_i \in J$ can be run only on a subset of the machines $S_i \subset M$. Is there an assignment of jobs to machines, such that the sum requirement of jobs assigned to each machine M_j equals 1?

For a given instance of P-NON-UNIFORM-LOAD-BALANCING we construct an instance of P-UNSPLITTABLE-ROUTING as follows (Fig. 7). Let $T = 1$, and $c_{\text{st}} = c_{\text{rt}} = 1$. Create a node J_i for each job $J_i \in J$, a node M_j for each machine $M_j \in M$, and add an edge (J_i, M_j) if $M_j \in S_i$. Connect all the nodes $M_j \in M$ to the sink. Let available energies at the nodes be $b_{J_i} = r_i$, $b_{M_j} = 2$.

Suppose that the instance of P-NON-UNIFORM-LOAD-BALANCING is a "yes" instance, i.e., there is an assignment of jobs to machines such that the sum requirement of jobs assigned to each machine equals 1. Observe the following rate assignment: $\lambda^* = \{\lambda_{J_i} = r_i, \lambda_{M_j} = 1\}$. This rate assignment is feasible only for the unsplittable routing in which M_j's descendants are the jobs assigned to M_j in the solution for P-NON-UNIFORM-LOAD-BALANCING. Moreover, as in this rate assignment all the nodes spend all their available energies and since $\sum_{i=1}^{k} b_{J_i} = \sum_{i=1}^{k} r_i = n$, it is not hard to see that this is the lexicographically maximum rate assignment that can be achieved for any instance of P-NON-UNIFORM-LOAD-BALANCING. If the instance of P-NON-UNIFORM-LOAD-BALANCING is a "no" instance, then P-UNSPLITTABLE-ROUTING at the optimum necessarily produces a rate assignment that is lexicographically smaller than λ^*.

Therefore, if P-Unsplittable-Routing can be solved in polynomial time, then P-Non-uniform-Load-Balancing can also be solved in polynomial time. □

As the proof of Lemma 14 is constructed for $T = 1$, it follows that P-Unsplittable-Routing is NP-hard for general T, in either time-variable or time-invariable setting.

On the other hand, determining a time-invariable unsplittable routing that guarantees the maximum value of the minimum sensing rate over all time-invariable unsplittable routings is solvable in polynomial time, and we provide a combinatorial algorithm that solves it below.

We first observe that in any time-invariable unsplittable routing, if all the nodes are assigned the same sensing rate λ, then every node i spends a fixed amount of energy Δb_i per time slot equal to the energy spent for sensing and sending own flow and for forwarding the flow coming from the descendant nodes: $\Delta b_i = \lambda \left(c_{\mathrm{st}} + c_{\mathrm{rt}} D_{i,t}\right)$.

The next property we use follows from the integrality of the max flow problem with integral capacities (see, e.g., [1]). This property was stated as a theorem in [20] for single-source unsplittable flows, and we repeat it here for the equivalent single-sink unsplittable flow problem:

Theorem 2 [20] *Let $G = (N, E)$ be a given graph with the predetermined sink s. If the supplies of all the nodes in the network are from the set $\{0, \lambda\}$, $\lambda > 0$, and the capacities of all the edges/nodes are integral multiples of λ, then: if there is a fractional flow of value f, there is an unsplittable flow of value at least f. Moreover, this unsplittable flow can be found in polynomial time.*

Note 2 For the setting of Theorem 2, any augmenting-path or push-relabel based max flow algorithm produces a flow that is unsplittable, as a consequence of the integrality of the solution produced by these algorithms. We will assume that the used max-flow algorithm has this property.

The last property we need is that our problem can be formulated in the setting of Theorem 2. We observe that for a given sensing rate λ, each node spends $c_{\mathrm{st}}\lambda$ units of energy for sensing, whereas the remaining energy can be used for routing the flow originating at other nodes. Therefore, for a given λ, we can set the supply of each node i to λ, set its capacity to $u_i = (\Delta b_i - c_{\mathrm{st}}\lambda)/c_{\mathrm{rt}}$ (making sure that $\Delta b_i - c_{\mathrm{st}}\lambda \geq 0$), and observe the problem as the feasible flow problem. For any feasible *unsplittable* flow solution with all the supplies equal to λ, we have that flow through every edge/node equals the sum flow of all the routing paths that contain that edge/node. As every path carries a flow of value λ, the flow through every edge/node is an integral multiple of λ. Therefore, to verify whether it is feasible to have a sensing rate of λ at each node, it is enough to down-round all the nodes' capacities to the nearest multiple of λ: $u_i = \lambda \cdot \lfloor (\Delta b_i - c_{\mathrm{st}}\lambda)/(c_{\mathrm{rt}}\lambda) \rfloor$, and apply the Theorem 2.

An easy upper bound for λ is $\lambda_{\max} = \min_i \Delta b_i / c_{\mathrm{st}}$, which follows from the battery nonnegativity constraint. The algorithm becomes clear now:

Lemma 15 Maxmin-Unsplittable-Routing *runs in time*

$$O(\log(\max_i (b_{i,1} + e_{i,1})/(c_{st}\delta))(MF(n+1, m))),$$

Algorithm 5 MAXMIN-UNSPLITTABLE-ROUTING(G, b, e)

```
1: Perform a binary search for $\lambda \in [0, \lambda_{\max}]$.
2: For each $\lambda$ chosen by the binary search set node supplies to $\lambda$ and node capacities to $u_i = \lambda \cdot \lfloor (\Delta b_i - c_{\mathrm{st}}\lambda)/(c_{\mathrm{rt}}\lambda) \rfloor$. Solve feasible flow problem.
3: Return the maximum feasible $\lambda$.
```

where $MF(n, m)$ is the running time of a max-flow algorithm on an input graph with n nodes and m edges.

6.2 Routing Tree

If it was possible to find the (either time variable or time-invariable) max-min fair routing tree in polynomial time for any time horizon T, then the same result would follow for $T = 1$. It follows that if P-ROUTING-TREE NP-hard for $T = 1$, it is also NP-hard for any $T > 1$. Therefore, we restrict our attention to $T = 1$.

Assume w.l.o.g. $e_{i,1} = 0\ \forall i \in V\backslash\{s\}$. Let $\mathcal{T}$ denote a routing tree on the given graph G, and $D_i^{\mathcal{T}}$ denote the number of descendants of a node i in the routing tree $\mathcal{T}$. Maximization of the common rate $\lambda_i = \lambda$ over all routing trees can be stated as:

$$\max_{\mathcal{T}} \min_{i \in N} b_i/(c_{\mathrm{st}} + c_{\mathrm{rt}} D_i^{\mathcal{T}}) \tag{26}$$

This problem is equivalent to maximizing the network lifetime for $\lambda_i = 1\ \forall i \in V\backslash\{s\}$ as studied in [5]. This problem, which we call P-MAXIMUM-LIFETIME-TREE, was proved to be NP-hard in [5] using a reduction from the SET-COVER problem [19]. The instance used in [5] for showing the NP-hardness of the problem has the property that the equivalent problem of finding a tree with the lexicographically maximum rate assignment, P-ROUTING-TREE, is such that at the optimum $\lambda_1 = \lambda_2 = \cdots = \lambda_n = \lambda$. Therefore, P-ROUTING-TREE is also NP-hard.

We will strengthen the hardness result here and show that the lower bound on the approximation ratio for the P-ROUTING-TREE problem is $\Omega(\log n)$, unless $P = NP$. Notice that because we are using the instance for which at the optimum $\lambda_i = \lambda\ \forall i$, the meaning of the approximation ratio is clear. In general, the optimal routing tree can have a rate assignment with distinct values of the rates, in which case we would need to consider an approximation to a vector $\{\lambda_i\}_{i \in \{1,\dots,n\}}$. However, we note that for any reasonable definition of approximation (e.g., element-wise or prefix-sum as in [21]) our result for the lower bound is still valid. As for the instance we use P-ROUTING-TREE is equivalent to the P-MAXIMUM-LIFETIME-TREE problem, the lower bound applies to both problems.

We extend the reduction from the SET-COVER problem used in [5] to prove the lower bound on the approximation ratio. In the SET-COVER problem, we are given elements $1, 2, \dots, n^*$ and sets $S_1, S_2, \dots, S_m \subset \{1, 2, \dots, n^*\}$. The goal is to determine the minimum number of sets from $S_1, \dots, S_m$ that cover all the elements $\{1, \dots, n^*\}$. Alternatively, the problem can be recast as a decision problem that determines whether there is a set cover of size k or not. Then the minimum set cover can be determined by finding the smallest k for which the answer is "yes".

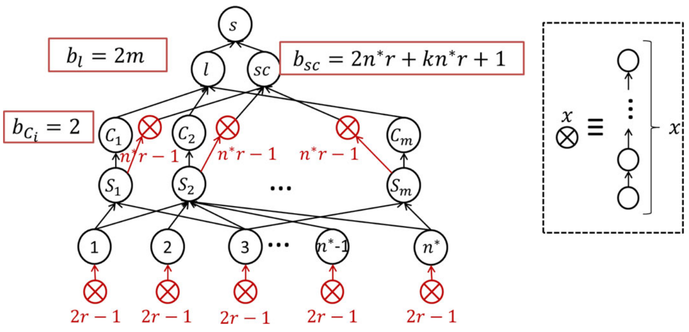


Fig. 8 A lower bound on the approximation ratio for P-ROUTING-TREE. Nodes $1, 2, \ldots, n^*$ correspond to the elements, whereas nodes $S_1, S_2, \ldots, S_m$ correspond to the sets in the SET-COVER problem. An element node i is connected to a set node S_j if in the SET-COVER problem $i \in S_j$. If there is a set cover of size k, then at $\lambda = 1$ all the non-set-cover nodes are connected to the tree rooted at the node l, whereas all the set cover nodes and all the element nodes are in the tree rooted at sc. The line-topology graphs represented by *crossed circles* are added to limit the size of an approximate solution to the SET-COVER problem

Suppose that there exists an approximation algorithm that solves P-ROUTING-TREE (or P-MAXIMUM-LIFETIME-TREE) with the approximation ratio r. For a given instance of SET-COVER, construct an instance of P-ROUTING-TREE as in Fig. 8 and denote it by G. This reduction is similar to the reduction used in [5], with modifications being made by adding line-topology graphs, and by modifying the node capacities appropriately to limit the size of the solution to the corresponding SET-COVER problem. Let l^x denote a directed graph with line topology of size x. Assume that all the nodes in any l^x have capacities that are non-constraining. By the same observations as in the proof of NP-completeness of P-MAXIMUM-LIFETIME-TREE [5], if there is a routing tree that achieves $\lambda = 1$, then there is a set cover of size k for the given input instance of SET-COVER.

Now observe a solution that an approximation algorithm with the ratio r would produce, that is, an algorithm for which $\frac{1}{r} \leq \lambda \leq 1$ when $\lambda_{\text{OPT}} = 1$.

Lemma 16 *In any routing tree for which $\frac{1}{r} \leq \lambda \leq 1$, each node C_j can have at most one descendant.*

Proof Suppose that there is some routing tree T in which some C_j, $j = \{1, \ldots, m\}$ has more than 1 descendants. Then C_j must have at least one element node as its descendant. But if C_j has an element node as its descendant, then the line-topology graph connected to that element node must also be in C_j's descendant list, because T must contain all the nodes, and a line-topology graph connected to the element node has no other neighbors. Therefore, C_j has at least $2r + 1$ descendants. If $\lambda \geq \frac{1}{r}$, then the energy consumption at node C_j is $\frac{2r+2}{r} > 2$. But the capacity of the node C_j is 2, which is strictly less than the energy consumption; therefore, a contradiction. □

Lemma 16 implies that if there is a routing tree that achieves $\frac{1}{r} \leq \lambda \leq 1$, then all the element nodes will be connected to the tree rooted at sc through the set nodes they belong to. Therefore, the subtree rooted at sc will correspond to a set cover. The next question to be asked is how large can this set cover be (as compared to k)? The next lemma deals with this question.

Lemma 17 *If there is a routing tree $\mathcal{T}$ that achieves $\frac{1}{r} \leq \lambda \leq 1$, then the subtree rooted at sc in $\mathcal{T}$ contains at most $p \leq 3rk$ nodes.*

Proof Let $\mathcal{T}$ be a routing tree that achieves $\frac{1}{r} \leq \lambda \leq 1$.

The capacity of the node sc determines the number of the set nodes that can be connected to sc. As all the element nodes (and line-topology graphs connected to them) are in the subtree rooted at sc, when there are p set nodes connected to sc, sc has $2n^*r + pn^*r$ descendants. As each node has $\frac{1}{r} \leq \lambda \leq 1$ sensing rate, the energy consumption at the node sc is $e_{sc} = (2n^*r + pn^*r + 1)\lambda$. For the solution to be feasible, it must be $e_{sc} \leq b_{sc}$. Therefore:

$$\begin{aligned}
&(2n^*r + pn^*r + 1)\lambda \leq 2n^*r + kn^*r + 1 \\
&\Leftrightarrow \ p \leq \frac{1}{\lambda} \cdot \frac{2n^*r + kn^*r + 1}{n^*r} - \frac{2n^*r + 1}{n^*r} \\
&= \frac{1}{\lambda}\left(2 + k + \frac{1}{n^*r}\right) - 2 - \frac{1}{n^*r}
\end{aligned}$$

As $\lambda \geq \frac{1}{r}$: $p \leq (2+k)r + \frac{1}{n^*} - 2 - \frac{1}{n^*r} \leq (2+k)r \leq k \cdot 3r$, where the last inequality comes from $k \geq 1$. □

The last lemma implies that if we knew how to solve P-ROUTING-TREE in polynomial time with the approximation ratio r, then for an instance of SET-COVER we could run this algorithm for $k = \{1, 2, \ldots, m-1\}$ (verifying whether $k = m$ is a set cover is trivial) and find a $3r$-approximation for the minimum set cover, which is stated in the following lemma.

Lemma 18 *If there is a polynomial-time r-approximation algorithm for* P-ROUTING-TREE, *then there is a polynomial-time $3r$-approximation algorithm for* SET-COVER.

Proof Suppose that there was an algorithm that solves P-ROUTING-TREE in polynomial time with some approximation ratio r. For a given instance of SET-COVER construct an instance of P-ROUTING-TREE as in Fig. 8. Solve (approximately) P-ROUTING-TREE for $k \in \{1, \ldots, m-1\}$. In all the solutions, it must be $\lambda \leq 1$. Let k_m denote the minimum $k \in \{1, \ldots, m-1\}$ for which $\lambda \geq \frac{1}{r}$. Then the minimum set cover size for the input instance of SET-COVER is $k^* \geq k_m$, otherwise there would be some other $k'_m < k_m$ for which $\lambda \geq \frac{1}{r}$. From Lemmas 16 and 17, the solution to the constructed instance of P-ROUTING-TREE corresponds to a set cover of size $p \leq 3r \cdot k_m$ for the input instance. But this implies $p \leq 3r \cdot k^*$, and, therefore, the algorithm provides a $3r$-approximation to the SET-COVER. □

Theorem 3 *It is NP-hard to approximately solve* P-ROUTING-TREE *with an approximation ratio better than $\Omega(\log n)$.*

Proof The lower bound on the approximation ratio of SET-COVER was shown to be $\Omega(\log n)$ in [25].

Table 3 Running times of the algorithms for the WATER-FILLING-FRAMEWORK implementation

Problem	Running time
P-DETERMINE-RATES	$O(nT(nT\log(\frac{B+\max_{i,t} e_{i,t}}{(\delta c_{\mathrm{st}})}) + mT))$
P-FIXED-FRACTIONAL-ROUTING	$O(n\log(\frac{b_{i,1}+e_{i,1}}{\delta})(T + MF(n,m)))$
P-FRACTIONAL-ROUTING	$\widetilde{O}(nT(T^2\epsilon^{-2} \cdot (nT + MCF(n,m) + LP(mT, nT)))$

The proof for the lower bound on the approximation ratio given in [25] was derived assuming a polynomial relation between n^* and m. Therefore, the lower bound of $\Omega(\log n^*)$ holds for $m = n^{*c^*}$, where $c^* \in \mathbb{R}$ is some positive constant. Assume that $n^* \geq 3$. The graph given for an instance of SET-COVER (as in Fig. 8) contains $n = 2rn^* + mrn^* + 3 \leq rn^{*c'}$ nodes, for some other constant $c' > 1$. Therefore: $n^* \geq \sqrt[c']{\frac{n}{r}}$. As $r \geq \frac{1}{3}c\log n^*$, it follows that:

$$\begin{aligned} r &\geq \frac{1}{3}c\log\sqrt[c']{\frac{n}{r}} = \frac{c}{3c'}(\log n - \log r) \\ \Leftrightarrow \quad \frac{c}{3c'}\log r + r &\geq \frac{c}{3c'}\log n \Rightarrow r \geq c''\log n, \end{aligned}$$

for some $c'' \in \mathbb{R}$. □

7 Conclusions and Future Work

This paper presents a comprehensive algorithmic study of the max-min fair rate assignment and routing problems in energy harvesting networks with predictable energy profiles. We develop algorithms for the WATER-FILLING-FRAMEWORK implementation under various routing types. The running times of the developed algorithms are summarized in Table 3. The algorithms provide important insights into the structure of the problems, and can serve as benchmarks for evaluating distributed and approximate algorithms possibly designed for unpredictable energy profiles.

The results reveal interesting trade-offs between different routing types. For example, while we provide an efficient algorithm that solves the rate assignment in any routing specified at the input, we also show that determining a routing with the lexicographically maximum rate assignment for any routing tree or an unsplittable routing is NP-hard. On the positive side, we are able to construct a combinatorial algorithm that determines a time-invariable unsplittable routing which maximizes the minimum sensing rate assigned to any node in any time slot.

Fractional time-variable routing provides the best rate assignment (in terms of lexicographic maximization), and both the routing and the rate assignment are determined jointly by one algorithm. However, as demonstrated in Sect. 4, the problem is unlikely to be solved optimally without the use of linear programming, incurring a high running time. While we provide an FPTAS for this problem, reducing the algorithm running time by a factor of $O(nT)$ (as compared to the framework of [8,24,31]), the proposed algorithm still requires solving $O(nT)$ linear programs.

If fractional routing is restricted to be time-invariable and with constant rates, the problem can be solved by a combinatorial algorithm, which we provide in Sect. 5. However, as discussed in the introduction, constant sensing rates often result in the underutilization of the available energy.

There are several directions for future work. For example, extending the model to incorporate the energy consumption due to the control messages exchange would provide a more realistic setting. Moreover, designing algorithms for unpredictable energy profiles that can be implemented in an online and/or distributed manner is of high practical significance.

Acknowledgments The authors are grateful to Mihalis Yannakakis and David Johnson for useful discussions.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice-Hall Inc, Upper Saddle River, NJ (1993)
2. Bacinoglu, B., Uysal-Biyikoglu, E.: Finite-horizon online transmission rate and power adaptation on a communication link with Markovian energy harvesting. (2013). ArXiv preprint, http://arxiv.org/abs/1305.4558
3. Bertsekas, D., Gallager, R.: Data Networks, 2nd edn. Prentice-Hall Inc, Upper Saddle River, NJ (1992)
4. Blasco, P., Gunduz, D., Dohler, M.: A learning theoretic approach to energy harvesting communication system optimization. In: Proceedings of IEEE Globecom Workshops'12 (2012)
5. Buragohain, C., Agrawal, D., Suri, S.: Power aware routing for sensor databases. In: Proceedings of IEEE INFOCOM'05 (2005)
6. Chang, J.H., Tassiulas, L.: Maximum lifetime routing in wireless sensor networks. IEEE/ACM Trans. Netw. **12**(4), 609–619 (2004)
7. Charny, A., Clark, D.D., Jain, R.: Congestion control with explicit rate indication. In: Proceedings of IEEE ICC'95 (1995)
8. Chen, S., Fang, Y., Xia, Y.: Lexicographic maxmin fairness for data collection in wireless sensor networks. IEEE Trans. Mob. Comput. **6**(7), 762–776 (2007)
9. Chen, S., Sinha, P., Shroff, N., Joo, C.: Finite-horizon energy allocation and routing scheme in rechargeable sensor networks. In: Proceedings of IEEE INFOCOM'11 (2011)
10. Chen, S., Sinha, P., Shroff, N., Joo, C.: A simple asymptotically optimal energy allocation and routing scheme in rechargeable sensor networks. In: Proceedings of IEEE INFOCOM'12 (2012)
11. DeBruin, S., Campbell, B., Dutta, P.: Monjolo: an energy-harvesting energy meter architecture. In: ACM SenSys'13 (2013)
12. Gatzianas, M., Georgiadis, L., Tassiulas, L.: Control of wireless networks with rechargeable batteries. IEEE Trans. Wirel. Commun. **9**(2), 581–593 (2010)
13. Gorlatova, M., Bernstein, A., Zussman, G.: Performance evaluation of resource allocation policies for energy harvesting devices. In: Proceedings of WiOpt'11 (2011)
14. Gorlatova, M., Kinget, P., Kymissis, I., Rubenstein, D., Wang, X., Zussman, G.: Challenge: ultra-low-power energy-harvesting active networked tags (EnHANTs). In: Proceedings of ACM MobiCom'09 (2009)
15. Gorlatova, M., Margolies, R., Sarik, J., Stanje, G., Zhu, J., Vigraham, B., Szczodrak, M., Carloni, L., Kinget, P., Kymissis, I., Zussman, G.: Energy harvesting active networked tags (EnHANTs): prototyping and experimentation. Technical Report 2012-07-27, Columbia University (2012)
16. Gorlatova, M., Wallwater, A., Zussman, G.: Networking low-power energy harvesting devices: measurements and algorithms. IEEE Trans. Mob. Comput. **12**(9), 1853–1865 (2013)

17. Gurakan, B., Ozel, O., Yang, J., Ulukus, S.: Energy cooperation in energy harvesting two-way communications. In: Proceedings of IEEE ICC'13 (2013)
18. Huang, L., Neely, M.: Utility optimal scheduling in energy-harvesting networks. IEEE/ACM Trans. Netw. **21**(4), 1117–1130 (2013)
19. Karp, R.: Reducibility among combinatorial problems. In: Miller, R., Thatcher, J. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press, New York (1972)
20. Kleinberg, J.: Single-source unsplittable flow. In: Proceedings of IEEE FOCS'96 (1996)
21. Kleinberg, J., Rabani, Y., Tardos, E.: Fairness in routing and load balancing. In: Proceedings of IEEE FOCS'99 (1999)
22. Lenstra, J., Shmoys, D., Tardos, E.: Approximation algorithms for scheduling unrelated parallel machines. Math. Program. **46**(1–3), 259–271 (1990)
23. Lin, L., Shroff, N., Srikant, R.: Asymptotically optimal energy-aware routing for multihop wireless networks with renewable energy sources. IEEE/ACM Trans. Netw. **15**(5), 1021–1034 (2007)
24. Liu, R.S., Fan, K.W., Zheng, Z., Sinha, P.: Perpetual and fair data collection for environmental energy harvesting sensor networks. IEEE/ACM Trans. Netw. **19**(4), 947–960 (2011)
25. Lund, C., Yannakakis, M.: On the hardness of approximating minimization problems. J. ACM **41**(5), 960–981 (1994)
26. Madan, R., Lall, S.: Distributed algorithms for maximum lifetime routing in wireless sensor networks. IEEE Trans. Wirel. Commun. **5**(8), 2185–2193 (2006)
27. Mao, Z., Koksal, C., Shroff, N.: Near optimal power and rate control of multi-hop sensor networks with energy replenishment: Basic limitations with finite energy and data storage. IEEE Trans. Autom. Control **57**(4), 815–829 (2012)
28. Megiddo, N.: Optimal flows in networks with multiple sources and sinks. Math. Program. **7**(1), 97–107 (1974)
29. Ozel, O., Tutuncuoglu, K., Yang, J., Ulukus, S., Yener, A.: Transmission with energy harvesting nodes in fading wireless channels: optimal policies. IEEE J. Sel. Areas Commun. **29**(8), 1732–1743 (2011)
30. Plotkin, S., Shmoys, D., Tardos, E.: Fast approximation algorithms for fractional packing and covering problems. Math. Oper. Res. **20**(2), 257–301 (1995)
31. Radunović, B., Boudec, J.Y.L.: A unified framework for max-min and min-max fairness with applications. IEEE/ACM Trans. Netw. **15**(5), 1073–1083 (2007)
32. Sarkar, S., Khouzani, M., Kar, K.: Optimal routing and scheduling in multihop wireless renewable energy networks. IEEE Trans. Autom. Control **58**(7), 1792–1798 (2013)
33. Sarkar, S., Tassiulas, L.: Fair allocation of discrete bandwidth layers in multicast networks. In: Proceedings of IEEE INFOCOM'00 (2000)
34. Srivastava, R., Koksal, C.: Basic performance limits and tradeoffs in energy-harvesting sensor nodes with finite data and energy storage. IEEE/ACM Trans. Netw. **21**(4), 1049–1062 (2013)