Data Aggregation Tree Construction Strategies for Increasing Network Lifetime in EH-WSN

Conference Paper · December 2015

DOI: 10.1109/GLOCOM.2015.7417049

3 authors:

Miloud Bagaa
Aalto University, School of electrical engineering
119 PUBLICATIONS **2,775** CITATIONS
SEE PROFILE

Mohamed Younis
University of Maryland, Baltimore County
355 PUBLICATIONS **16,557** CITATIONS
SEE PROFILE

Ilangko Balasingham
Oslo University Hospital & Norwegian University of Science and Technology
317 PUBLICATIONS **4,024** CITATIONS
SEE PROFILE

Data aggregation tree construction strategies for increasing network lifetime in EH-WSN

Miloud Bagaa*, Mohamed Younis§ and Ilangko Balasingham*‖
* Dep. of Ele. and Tel., NTNU, 7491 Trondheim, Norway. Emails:{miloudb,ilangkob}@iet.ntnu.no
§ Dep. of Comp Sc. & Elect Eng, Univ. of Maryland Baltimore County. Email:younis@cs.umbc.edu
‖ Intervention Center, Oslo University Hospital and Institute of Clinical Medicine, University of Oslo,0372 Oslo, Norway

***Abstract*—Energy scavenging from ambient sources represents a promising solution for sustaining continual operation for wireless sensor networks. An energy harvesting wireless sensor network (*EH-WSN*) can have two node types, harvesting enabled nodes (*HNs*) and non-harvesting enabled nodes (*NHNs*). In this paper, we consider the problem of achieving network longevity in *EH-WSN* when in-network data aggregation is used. The aim is to construct an aggregation tree that extends the network lifetime through reducing the overhead on NHN as much as possible. Two solutions are proposed; the first model the problem using a integer linear program, whereas the second solution uses minimum directed spanning tree. The objective of these protocols is to extend the network lifetime while reducing the runtime complexity. Both solutions are evaluated through extensive simulation experiments. The obtained results demonstrate their feasibility and ability in achieving the design goals.**

***Index Terms*—wireless sensor network, energy harvesting, in-network data aggregation.**

I. Introduction

Wireless sensor networks (WSNs) can be employed in harsh environments for monitoring serious events, such as forest fire and toxic gas leakage. Since a *WSN* is often deployed in a wide and inhospitable area replacing the battery of sensor nodes impractical, and may be even infeasible. For this reason, energy harvesting from ambient sources has become an appealing option for sustaining an extended network lifetime. Wireless communication is the most energy consumer in a sensor node. Given the multi-hop routing topology, nodes that serve as relays in the network suffer from increased load and consequently higher rate of energy depletion than data sources. When a data aggregation tree is used, the lifetime of a node depends on number of its children. Usually, the data are aggregated and forwarded within a tree structure. Most of the existing protocols assume that the received data can be aggregated into one packet. This can be justified based on the observation that many aggregation functions, like min, max and $average$, are many-to-one in nature and the results can be transmitted in just one packet.

Most published studies on *EH-WSN* have been devoted to stochastic harvesting models, [1] and [2], that work at MAC layer to design an optimal strategy for forwarding the received packets while extending the network lifetime. Other work, e.g., [3] and [4], has focused on the network layer to optimally manage the harvesting and the forwarding strategies for extending the network lifetime. Unlike prior work, in this paper we consider the formation of a data aggregation tree that best suits EH-WSN. In such a network model, two types of sensor nodes may exist, namely harvesting enabled nodes (HN), and non-harvesting nodes (NHN). To the best of our knowledge, this problem has not be studied in the literature. We propose two solutions. The first, which is called Packet payload Aggregation for increased Lifetime (*PAL*), uses a integer linear program formulation to find the optimal aggregation tree for extending the network lifetime in EH-WSN. However, *PAL* takes long to execute especially for a large network. In order to overcome this limitation, the second solution, named accelerated *PAL* (*APAL*), that uses minimum directed spanning tree to decrease the runtime complexity. The performance of the purposed solutions are validated through simulation.

The rest of the paper is organized as follows. In Section II, we formally define the problem. *PAL* solution is discussed in detail in section III. Section IV describes *APAL*. Section V presents the simulation results. The related work is summarized in section VI. Finally, section VII concludes the paper.

II. Background and network model

A. Definition

Definition 1. *A weighted directed graph $dG = (V, E, \omega)$ consists of a set of vertices V and a set of weighted directed edges E. ω represents the weight of these edges, $\omega_{i,j} \in \omega$ denote the weight of edge $(i, j) \in E$. Since the edges of a directed graph are represented with ordered vertex pairs, $\omega_{i,j}$ may be different from $\omega_{j,i}$.*

Definition 2. *A directed spanning tree (DST) of a weighted directed graph dG rooted at S, is a subgraph T of dG that contains a directed path from S to every other vertex in V. Indeed, the subgraph T should not contain any cycle. The weight $W(T)$ of a directed spanning tree T is the sum of the weight of its edges, i.e., $W(T) = \sum\limits_{(i,j)\in T} \omega_{i,j}$. A minimum directed spanning tree (MDST) is the one that has the least weight.*

B. Network model

We consider a network of N stationary nodes that is modeled as a graph $G(V, H, E)$, where the set of vertices V represents the nodes, with a base-station (sink node) $S \in V$, and the set of edges E that represents the neighborhood

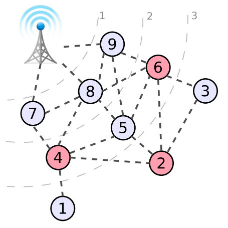


Fig. 1: An example *EH-WSN* topology

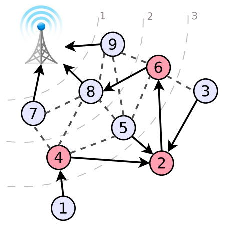


Fig. 2: Results of *PAL* execution on the topology of Fig. 1

relation between nodes. An edge $(i, j) \in E$ means that nodes i and j have a communication link. We assume that a subset of nodes, $H \subseteq V$, have the ability to harvest energy from the environment. In other words, H consists of *HN* nodes, whereas the members $V \setminus \{H \cup S\}$ are *NHN* nodes. We assume that the HN nodes can scavenge sufficient energy to sustain uninterrupted operation. Thus, the nodes in H can be viewed as energy unconstrained nodes. To clarify the notation, Fig. 1 shows a network topology of 9 nodes and base-station. The dotted lines between nodes denote connectivity, i.e., the presence of a communication link between a pair of nodes. In this Figure, three nodes, which are 2, 4 and 6 are HN, whereas the others are NHN.

C. Problem formulation

In this paper, we propose a strategy for gathering and aggregating data from all N nodes to the base-station while extending the network lifetime. As defined in [5], the network lifetime is the duration starting from the network setup to the time when the first sensor node runs out of energy. We assume a many-to-one aggregation function, like $\min$ and $\max$, such that each node can process the data payload of the received packets from its children and generate only one packet. In this case, a node turns its radio on only when it has to transmit data to its parent or when it has to receive data from its children. Therefore, the network lifetime as defined above is a decreasing function of the maximum number of children that any node in $V \setminus \{H \cup S\}$ can have in the network. To extend the network lifetime, we aim to construct a data aggregation tree that reduces the number of children for nodes in $V \setminus \{H \cup S\}$.

III. Data Aggregation for Increased Lifetime of EH-WSN

As pointed out above, the network lifetime depends on the *NHN* node that has the highest number of children. In this section, we present our optimization approach for Packet payload Aggregation for increased Lifetime (*PAL*). *PAL* models the aggregation tree formation in EH-WSN as a integer linear program, that aims to increase the EH-WSN lifetime by reducing the number of children of NHN nodes in $V \setminus \{H \cup S\}$. Let w_i denotes the weight of node i and is defined as follows:

$$w_i = \begin{cases} 0 & i \in H \cup \{S\} \\ 1 & \text{Otherwise} \end{cases}$$

The *PAL* formulation involves two variables: (i) $X_{i,j}$ is a decision boolean variable that is set to 1 if node i selects node j as its parent; (ii) $F_{i,j}$ is a flow matrix of integers that enforces the optimization to yield a connected topology; in our case a tree that is rooted at the base-station S. To ensure that the constructed topology is a tree, we have to ensure that the entered flow to S equals to $N-1$ (a tree has $|V|-1$ edges). We denote by $\eta(i)$ the neighbors of node i in a graph G.

$$\mathbf{min} \max_{\forall j \in V-\{S\}} w_j \sum_{i \in \eta(j)} X_{i,j}$$

s.t,

$$\forall i \in V - \{S\}, \sum_{j \in \eta(i)} X_{i,j} = 1 \tag{1}$$

$$\sum_{i \in \eta(S)} \mathcal{F}_{i,S} = N - 1 \tag{2}$$

$$\forall i \in V - \{S\}, \sum_{j \in \eta(i)} \mathcal{F}_{i,j} - \sum_{j \in \eta(i)-\{S\}} \mathcal{F}_{j,i} = 1 \tag{3}$$

$$\forall (i,j) \in E, i \neq S, 0 \leq \mathcal{F}_{i,j} \leq (N-1)X_{i,j} \tag{4}$$

PAL is modeled as min-max problem, where we aim to minimize the maximum number of children of a node in NHN. Meanwhile, the constraints are used to ensure the following conditions: constraint (1) ensures that each node in the network has only one parent. Meanwhile, constraints (2), (3) and (4) are used for modeling the connectivity requirements. They ensure that the constructed topology is a tree, i.e., it is connected and rooted at the base-station S, and does not contains any cycle. To do so, it is sufficient to ensure that the constructed topology is connected and has $N-1$ directed edges. For this reason, we mimic packet flow generated from the network nodes to the base-station. Each relay node in the network has to forward to its parent the received packets from its children plus one, while the leaf nodes transmit only one packet. We have to guarantee that the generated packets is received by S and equals to $N-1$. We also enforce that the generated flow routed only on the constructed topology. Constraint (2) ensures that the number of edges used in the constructed topology equals to $N-1$. This captures the fact that the flow entered to the base-station equals exactly to the number of nodes. By constraint (3), each node in the network is forced to generate only one packet. The number of transmitted packets by a node i exceeds the received count by just one. Constraint (4) ensures that the generated flow should be routed only in the data aggregation tree. A node i forwards a packet only to its parent j in the constructed topology ($X_{i,j} = 0 \Rightarrow \mathcal{F}_{i,j} = 0$). From constraints (2), (3) and (4), we conclude that: (i) each node should generate only one packet; (ii) the number of received packets by the base-station equals to $N-1$. This enforces the constructed topology to be a tree. Again, the flow concept used in (2), (3) and (4) are just for modeling the connectivity requirements and do not reflect actual flow of data packets in the data aggregation tree. Fig. 2 shows the result of *PAL* execution on the topology presented in Fig. 1. In this example, all NHN nodes, except node 8, are leaf nodes, which would have a positive impact. Also, the number of children of node 8 is reduced to one in order to increase further the network lifetime.

Theorem 1. The complexity of *PAL* is more than $\mathcal{O}(16^{|E|})$.

Proof: Existing tools for solving the above integer linear program formulation, like CPLEX and Gurobi, use the branch-and-bound method. Jeroslow [6] has proven that the branch-and-bound for binary linear program has $\mathcal{O}(2^L)$ as runtime complexity, where L is the number of binary variable in the system. In the *PAL* model, we have $4|E|$ integer variables. Based on the observation, solving the integer linear program is more complicated than solving a binary linear program, and thus the runtime complexity of *PAL* is more than $\mathcal{O}(16^{|E|})$. ■

IV. Enhancing Lifetime through MDST

Based on Theorem 1, the computational complexity of *PAL* is exponential in the number of edges in the network. Despite yielding optimal solution, the integer linear program formulation does not scale well for large setups. In this section, we present an accelerated *PAL* (*APAL*) algorithm that has a polynomial runtime complexity. The idea behind APAL is to use MDST to form a data aggregation tree T that extends the network lifetime. The construction of MDST $\mathcal{T} = (V, E_{\mathcal{T}})$ for a directed graph $dG(V, \mathbb{E}, \omega)$ can be done in a polynomial time. MDST is proposed in literature to allow the root to communicate with all nodes with lowest cost. In contrast to MDST $\mathcal{T}$, in data aggregation tree T the packets are forwarded from the sources to the root. Indeed, a data aggregation tree T is a reverse directed spanning tree. *NHN* nodes can be prevented to be parents in T, when executing a MDST algorithm by assigning their incoming edges the highest weight. In the balance of this section, we will give first an overview on Edmond algorithm [7], which is used for forming a MDST. Then, we present an accelerated *PAL* (*APAL*) algorithm.

A. Edmond's algorithm

Let $dG(V, \mathbb{E}, \omega)$ be a directed weighted graph and S be the root of the desired MDST $\mathcal{T} = (V, E_{\mathcal{T}})$. Edmonds algorithm starts by removing all the directed edges incoming to the root S. Then, for each vertex in the graph, the incoming directed edge that has the smallest weight $\omega_{i,j}$ is selected and added to $E_{\mathcal{T}}$. If the constructed tree $\mathcal{T}$ does not contain any cycle, the algorithm finishes and $\mathcal{T}$ becomes the MDST. Otherwise, the algorithm identifies a cycle $C = \{(v_i, v_{i+1}), (v_{i+1}, v_{i+2}) \cdots (v_{i+k}, v_i)\}$ and removes it. Let $e = (u, v) \in \mathbb{E}$ be a directed edge, we denote by

Algorithm 1 Algorithm of APAL

```
Input:
      $G = (V, E, H)$: Network connectivity graph. $H$ represents the harvesting nodes.
      $S$: The base-station.
Output:
      $T = (V, \mathcal{E})$: Data aggregation tree, where $\mathcal{E}$ are the arcs of the tree.
 1: $W = \{\}$;
 2: for all $u \in V$ do
 3:     if $u \in H \cup \{S\}$ then
 4:         $W[u] = 0$;
 5:     else
 6:         $W[u] = 1$;
 7:     end if
 8: end for
    Construct a directed weighted graph $dG(V, \mathbb{E}, \omega)$:
 9: for all $(u, v) \in E$ do
10:     Transform $(u, v)$ into two directed edges $(u, v)$ and $(v, u)$ in $\mathbb{E}$;
11:     $\omega[u, v] = W[u]$;
12:     $\omega[v, u] = W[v]$;
13: end for
    Remove base-station's incident directed edges in $dG(V, \mathbb{E}, \omega)$
14: for all $(u, v) \in \mathbb{E}$ do
15:     if $v == S$ then
16:         remove $(u, v)$ from $\mathbb{E}$;
17:     end if
18: end for
    Construct minimum directed spanning tree $\mathcal{T} = (V, E_{\mathcal{T}})$ through Edmonds algorithm
19: $\mathcal{T}$ = Edmonds($dG(V, \mathbb{E}, \omega)$);
    Construct the data aggregation tree $T$ from $\mathcal{T}$
20: $\mathcal{E} = \{\}$;
21: for all $(u, v) \in E_{\mathcal{T}}$ do
22:     insert $(v, u)$ to $\mathcal{E}$
23: end for
24: Return $T = (V, \mathcal{E})$
```

$\pi(v) = u$ the source vertex of e. To remove the cycle C from $E_{\mathcal{T}}$, the algorithm contracts C into a single vertex v_C and recalculates edge weights going in and out of the cycle. The vertices of new graph $d\acute{G}$ are $V - C \cup \{v_C\}$, which ensures that $d\acute{G}$ does not contain the cycle C. Hence, the algorithm recursively calls itself on the new graph $d\acute{G}$. When MDST $\acute{\mathcal{T}} = (V, \acute{E_{\mathcal{T}}})$ is computed for $d\acute{G}$, then $\mathcal{T} = (V, E_{\mathcal{T}})$ would be constructed by breaking the cycle C. Let $(u, v_C) \in \acute{E_{\mathcal{T}}}$ be an incoming directed edge to vertex v_C in $\acute{\mathcal{T}}$. Let $(u, v) \in \mathbb{E}$ with $v \in C$ be the corresponding edge of $(u, v_C) \in \acute{E_{\mathcal{T}}}$ in graph G. To construct $\mathcal{T} = (V, E_{\mathcal{T}})$, Edmond algorithm replaces (u, v_C) by (u, v) in $E_{\mathcal{T}}$ and then breaks the cycle by removing $(\pi(v), v)$ from $E_{\mathcal{T}}$.

B. APAL description

In the balance of this section, we use Algorithm 1 to explain the functionality of *APAL*. Fig. 3 serves as a detailed example that will be referenced to illustrate the operation of *APAL*. In the figure, a dashed arrows indicates the directed edges in the weighted directed graph $dG = (V, E, W)$. The blue dashed arrows represent the directed edges with weight $= 1$, while the red ones are for those with weight $= 0$. The solid arrows indicates the links of the constructed tree (i.e., MDST or data aggregation tree).

APAL starts by constructing a weighted directed graph $dG(V, \mathbb{E}, \omega)$ from undirected graph $G(V, E, H)$ (Algorithm 1: lines $1-13$). Firstly, the weight of vertices are defined according to their type (Algorithm 1: lines $1-8$), such that the weights of *HN* nodes and the base-station S are set to 0, whereas the weight of ordinary nodes NHN have set to 1. We have given a small weight to the base-station and *HN* nodes in order to favorite the selection of these nodes as parents later when forming MDST. Then, $dG(V, \mathbb{E}, \omega)$ is constructed

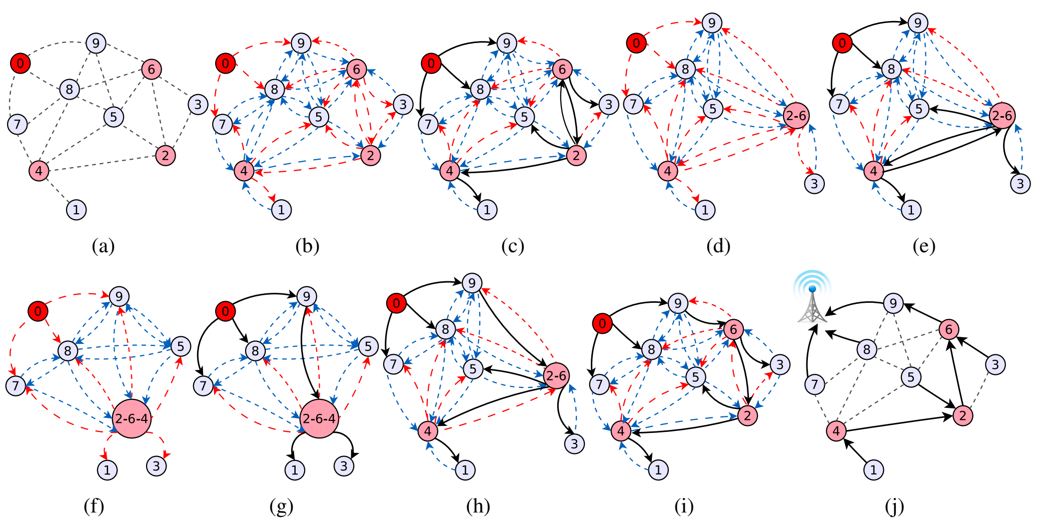


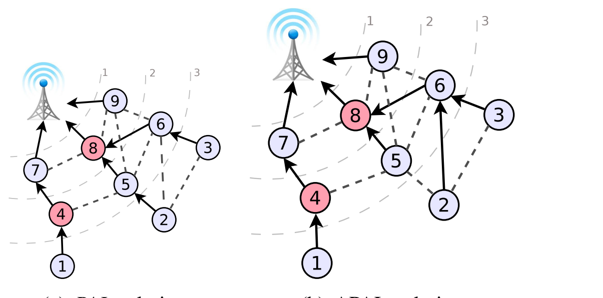
Fig. 3: Illustrative example that shows the execution of *APAL*

from $G(V,E,H)$ by transforming each edge $(u,v) \in E$ to two directed edges (u,v) and (v,u) in $\mathbb{E}$. The weight of a directed edge (u,v) is that of the source vertex u. In order to allow the execution of Edmond algorithm on $dG(V,\mathbb{E},\omega)$, all the incoming edges to the base-station will be removed from $\mathbb{E}$ (Algorithm 1: lines $14-18$). Figs $3(a)$ and $3(b)$ show how $dG(V,\mathbb{E},\omega)$ is constructed from $G(V,E,H)$.

Next, Edmonds algorithm is executed on $dG(V,\mathbb{E},\omega)$ to construct MDST (Algorithm 1: line 19). Figures $3(c)-(i)$ show the execution of MDST. As depicted in Fig. 3 (c), each vertex selects the incoming edges that has the smallest weight (i.e., the red dashed arrows) to MDST $\mathcal{T}=(V,E_{\mathcal{T}})$. However, a cycle $C_1 = \{(2,6),(6,2)\}$ is created in $E_{\mathcal{T}}$. To remove C_1 from $E_{\mathcal{T}}$, The algorithm contracts C_1 into a single vertex "2-6". Then, it recalculates edge weights going in and out of "2-6" as dpeicted in Fig. 3 (d). Fig. 3 (e) shows another iteration of the Edmond algorithm on the directed graph depicted in Fig. 3 (d). In Fig. 3 (e), another cycle is created $C_2 = \{(\text{“2-6”},4),(4,\text{“2-6”})\}$. Using the same approach, the cycle C_2 is removed in Fig. 3 (f). As depicted in Fig. 3 (g), $(9,\text{“2-6-4”}))$ is elected as incoming edge in "2-6-4". In Fig. 3 (h), $(9,\text{“2-6-4”}))$ is replaced with $(9,\text{“2-6”}))$ and $(4,\text{“2-6”})$ is removed, which leads to break the cycle C_2. For the same reason, cycle C_1 would be broken by replacing $(9,\text{“2-6”}))$ with $(9,6))$ and removing $(2,6)$ as depicted in Fig. 3 (i). At this point we have constructed MDST $\mathcal{T}=(V,E_{\mathcal{T}})$ as depicted in Fig. 3 (i). To construct the data aggregation tree $T=(V,\mathcal{E})$, we have to inverse the directed edges in $\mathcal{T}=(V,E_{\mathcal{T}})$. For every directed edge $(u,v) \in E_{\mathcal{T}}$ we create a new one (v,u) in $\mathcal{E}$ (Algorithm 1: lines $20-23$). Fig. 3 (j) shows the constructed data aggregation tree.

Theorem 2. APAL does not form an optimal data aggregation tree in EH-WSN.

Proof: We prove this theorem by a counter example. In the example depicted in Fig. 4, we have applied *APAL* and *PAL* to the same topology. In this case, the maximum number of children of a NHN node in the tree constructed by *PAL* is 1 (Fig. $4(a)$), whereas it is 2 for the one constructed by *APAL* (Fig. $4(b)$). Therefore, the constructed tree T cannot be the optimal data aggregation tree for EH-WSN. ∎



(a) *PAL* solution (b) APAL solution

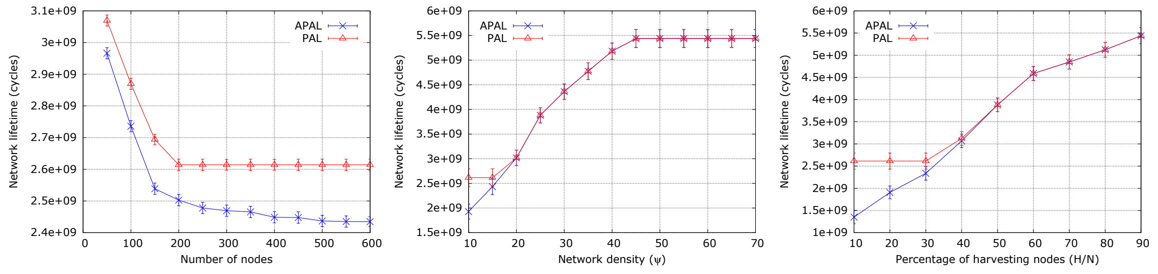
Fig. 4: Counter example which proves that *APAL* is not optimal

Theorem 3. The runtime complexity of the *APAL* algorithm is $\mathcal{O}(3|E|+2|V|(\log|V|+2))$.

Proof: We can subdivide the *APAL* algorithm into five steps. The first step corresponds to the loop (Algorithm 1: lines $2-8$), which iterates on $|V|$ and is thus $\mathcal{O}(|V|)$. The second and the third steps (Algorithm 1: lines $9-18$) are loops on $|E|$ and their complexity is $\mathcal{O}(2|E|)$. The fourth step is the execution of Edmond algorithm (Algorithm 1: line 19). Gabow et al. [8] implements Edmond algorithm with $\mathcal{O}(|E|\log|V|+|E|)$. Finally the Fifth step (Algorithm 1: line $21-23$), which has a runtime complexity of $\mathcal{O}(|V|-1) = \mathcal{O}(|V|)$. When considering all steps the complexity of *APAL* equals to $\mathcal{O}(3|E|+2|V|(\log|V|+2))$. ∎

Theorem 4. the network lifetime increases proportionally with the network density ψ and percentage of HN γ and inversely with the number of nodes N.

Proof: In *APAL* when a set of *NHN* nodes have one *HN* neighbor, all these nodes would select this neighbor as parent. This leads to increased network lifetime. Based on Fig. 4, a NHN node can be selected as a parent for more than



(a) Number of nodes (b) Network degree (c) Percentage of *HN* nodes (H/V)

Fig. 5: The network lifetime performance measured in terms of number of cycles until the first node runs out of energy

one node if and only if all its children do not have any *HN* neighbor. Therefore, the network lifetime is proportional to the probability that each node in the network has a *HN* neighbor. Let us assume that the nodes are deployed using uniform random distribution. Let ρ denotes the node transmission range. Thus, the communication coverage area of a node u is $C_u = \pi\rho^2$. The probability of node u has a neighbor is $P_{inside} = \frac{C_u}{\mathcal{A}}$, where $\mathcal{A}$ is the deployed area. Formally, we have $\psi = \frac{N}{\mathcal{A}}$, and then $P_{inside} = \frac{C_u \psi}{N}$. Let $P_u(HN)$ denote the probability that a node u has one *HN* neighbor. Then,

$$\begin{aligned} P_u(HN) &= \gamma P_{inside} \\ &= \frac{C_u \psi \gamma}{N} \end{aligned} \tag{5}$$

From equation 5, we conclude that the network lifetime is proportional to the network density ψ and percentage of HN nodes γ and inversely to the number of nodes N. ■

V. Simulation Results

The proposed solutions *PAL* and *APAL* are evaluated through simulation. The comparison is between *PAL* and *APAL* since no solutions in literature is proposed for constructing data aggregation tree that increases the network lifetime in EH-WSN. The algorithms are evaluated in terms of the following metrics: (i) Network lifetime, which is defined as the time to first battery drains out; (ii) Runtime, which is defined as the time needed to execute each solution. We have developed a simulator using Python and an extended package for graph theory called networkx [9]. *PAL* is implemented through IBM CPLEX tool. In the simulation results, each plotted point represents the average of 35 executions. The plots are presented with 95% confidence interval.

In our simulation, N nodes are deployed on a square area of length, D, according to a uniform random distribution. We generate a network topology for each N, and the network density $\psi = \frac{\Pi \times \rho^2 \times N}{D^2}$, where ρ is the node transmission range. We consider that ρ is fixed for all nodes, and hence the evaluation is performed by varying the number of nodes N, the network density ψ and the percentage of *HN* nodes $|H|/V$. We have conducted three types of experiments: (i) we vary N while fixing ψ to 15 and $|H|/V$ to 30%; (ii) We vary ψ while N is set to 300 and $|H|/V$ to 30%; (iii) We vary $|H|/V$ while we fix ψ to 15 and N to ψ to 15.

A. Network lifetime

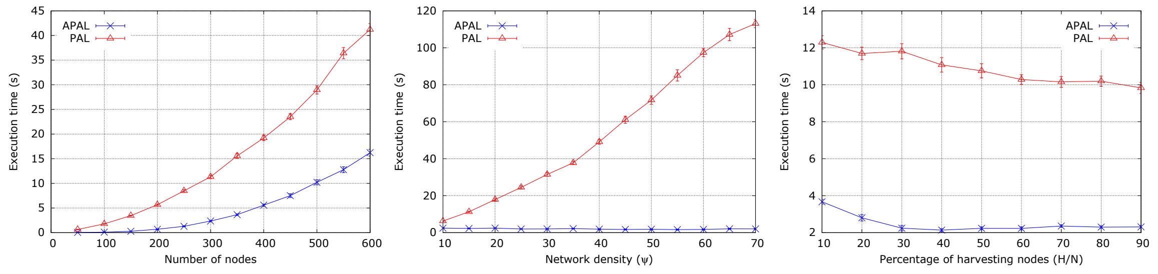
Fig. 5 shows the network lifetime as a function of N, ψ and $|H|/V$. The results clearly show that *PAL* outperforms *APAL* in terms of network lifetime, with the performance advantage growing with the increase of number of nodes. As depicted in Fig. 5(a), the gap between *PAL* and *APAL* expands from 3% when the number of nodes is 50 to 5% when the number of nodes up to 600. Figures 5(b) and 5(c) show that the gap in performance diminishes with the increase of ψ and $|H|/V$. When the network density exceeds 20 neighbors per node or the percentage of *HN* nodes exceed 40%, *PAL* and *APAL* would have the same performances in terms of network lifetime. This can be explained as follows. The increase of ψ leads to boosting the number of neighbors per node, which has a positive impact on the probability of selecting an *HN* parent. Growing the population of *HN* nodes has also a positive impact on the probability of selecting an *HN* parent. The increase of the number of nodes increases the probability that a node does not have *HN* neighbors, which has a negative impact on the network lifetime. The simulation results confirm what is stated in Theorem 4.

B. Runtime complexity

In contrast to *APAL* that is implemented through python, *PAL* is implemented via IBM CPLEX, which is powerful tool that uses multi-threading to speed-up the runtime execution. Fig. 6 shows the runtime complexity as a function of N, ψ and $|H|/V$. As indicated in the figure, the increase of the node count N and network density ψ pump the runtime complexity, whereas the increase of percentage of *HN* nodes has a positive impact. Overall, Fig. 6 shows that *APAL* outperforms *PAL* whatever the number of nodes, network density and the percentage of *HN* nodes. From Fig. 6(a), the difference in the running time between *APAL* and *PAL* increases sharply, where it reaches more than three times higher. Also, the gape in running time execution between *APAL* and *PAL* widens with the increase of network density ψ, where *PAL* takes up to 120 times, as depicted in Fig 6(b). Fig. 6(c) shows that whatever the percentage of *HN* nodes, *APAL* is better at least 10 times than *PAL* in terms of runtime execution.

VI. Related work

In-network data aggregation is a popular optimization methodology that helps to extend the network lifetime. Published data aggregation protocols use two kind of routing



(a) Number of nodes (b) Network degree (c) Percentage of *HN* nodes (H/V)

Fig. 6: Comparison of the execution time taken by both approaches

topology, namely tree-based and cluster-based, to forwarded the data from the sources to the base-station. TAG [10] is one of the most famous protocols that uses a tree structure, whereas, Improved-LEACH [11] is an example of those use a cluster-based approach. A survey can be found in [12]. In addition to network lifetime, a new category of data aggregation protocols has recently emerged which considers the quality of services when gathering and processing the sensor data [13]. Protocols in this category aim to reduce the data latency while preserving the network lifetime. A contention-free MAC protocol, e.g., TDMA (Time Division Multiple Access), is used achieve predicable and reduced time latency [14].

On the other hand, energy harvesting has been drawing lots of attention in recent years as a mean for sustain continual network operation. Many studies have been devoted to stochastic modeling of the harvesting events, e.g., availability of sunlight and high wind, where the objective is to adapt the packets forwarding rate to keep the network alive [1], [2]. Other work has focused on how to update the routing topology based on the energy scavenging rate. For more information, the reader can refer to the following survey [15]. Doost et al. [3] have developed a routing protocol where the nodes are charged wirelessly. A new metric is proposed for constructing the routes between the sources and destinations based on the charging of different nodes. Bin et al. [4] have considered the impact of wireless charging technology on *WSN* deployment. To the best of our knowledge, none of the published work considers the formation of an optimal data aggregation tree in energy harvesting wireless sensor network.

VII. Conclusion

In this paper, we have considered the problem of extending the network lifetime when data aggregation is used in energy harvesting wireless sensor networks (EH-WSN). The network is assumed to consist of two types of nodes, harvesting enabled nodes (*HNs*) and non harvesting enabled nodes (*NHNs*). To extend the network lifetime, we have presented two solutions that limit the selection of NHN nodes as parents. The first solution, named *PAL*, uses a integer linear program formulation, whereas the second one, named *APAL*, uses minimum directed spanning tree. *APAL* aims to reduce the runtime complexity at the cost of slight degradation of the solution optimality. The simulation results have demonstrated the effectiveness of proposed solutions for achieving their design goals. The results also have shown that *APAL* matches the lifetime performance of *PAL* as the percentage of NH nodes increases, yet at a fraction of the runtime complexity.

Acknowledgment This work was carried out during Dr. Bagaa's tenure of an ERCIM 'Alain Bensoussan' Fellowship Program.

References

[1] N. Michelusi and M. Zorzi, "Optimal random multiaccess in energy harvesting wireless sensor networks," in *IEEE ICC Workshops*, 2013.
[2] A. Cammarano, C. Petrioli, and D. Spenza, "Pro-energy: A novel energy prediction model for solar and wind energy-harvesting wireless sensor networks," in *IEEE MASS*, 2012, pp. 75–83.
[3] R. Doost, K. Chowdhury, and M. Di Felice, "Routing and link layer protocol design for sensor networks with wireless energy transfer," in *IEEE GLOBECOM'10*, December 2010, pp. 1–5.
[4] B. Tong, Z. Li, G. Wang, and W. Zhang, "How wireless power charging technology affects sensor network deployment and routing," in *IEEE ICDCS'10*, June 2010, pp. 438–447.
[5] S. Gandham, M. Dawande, R. Prakash, and S. Venkatesan, "Energy efficient schemes for wireless sensor networks with multiple mobile base stations," in *Proc. IEEE GLOBECOM'03*, December 2003, pp. 377–381.
[6] R. Jeroslow, "Trivial integer programs unsolvable by branch-and-bound," *Springer-Verlag Mathematical Programming*, vol. 6, no. 1, pp. 105–109, 1974. [Online]. Available: http://dx.doi.org/10.1007/BF01580225
[7] J. Edmonds, "Optimum branchings," *Journal of research of the Nation Bureau of Standards*, vol. 71B, no. 233-240, 1967.
[8] H. Gabow, Z. Galil, T. Spencer, and R. Tarjan, "Efficient algorithms for finding minimum spanning trees in undirected and directed graphs," *Springer-Verlag Combinatorica*, vol. 6, no. 2, pp. 109–122, 1986.
[9] Networkx, "http://networkx.lanl.gov."
[10] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks," *ACM SIGOPS Operating Syststems Review*, vol. 36, no. SI, pp. 131–146, 2002.
[11] W. Bo, H. Han-ying, and F. Wen, "An improved leach protocol for data gathering and aggregation in wireless sensor networks," in *IEEE ICCEE 2008*, Dec 2008, pp. 398–401.
[12] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *IEEE Wireless Commun*, vol. 14, no. 2, pp. 70–87, 2007.
[13] M. Bagaa, Y. Challal, A. Ksentini, A. Derhab, and N. Badache, "Data aggregation scheduling algorithms in wireless sensor networks: Solutions and challenges," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1339–1368, Third 2014.
[14] M. Bagaa, M. Younis, D. Djenouri, A. Derhab, and N. Badache, "Distributed low-latency data aggregation scheduling in wireless sensor networks," *ACM Transactions on Sensor Networks*, to appear.
[15] L. Xiao, P. Wang, D. Niyato, D. Kim, and Z. Han, "Wireless networks with rf energy harvesting: A contemporary survey," *IEEE Communications Surveys & Tutorials*, vol. PP, no. 99, pp. 1–1, 2015.