2015

Coverage problems in energy harvesting wireless sensor networks

Changlin Yang
University of Wollongong

Coverage Problems in Energy Harvesting Wireless Sensor Networks

A thesis submitted in partial fulfilment of the requirements for the award of the degree

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Changlin Yang

Bachelor of Engineering (Telecommunication)

School of Electrical, Computer and Telecommunications Engineering

April 2015

Statement of Originality

I, Changlin Yang, declare that this thesis, submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institutions.

Signed

Changlin Yang

April 27, 2015

Abstract

A Wireless Sensor Network (WSN), in general, contains many low-cost sensor nodes, each with the capability to monitor its surrounding environment. As conventional or non-rechargeable WSNs are restricted by the finite battery capacity of sensor nodes, researchers have sought the help of renewable energy techniques such as solar and wind. Consequently, each sensor node, in the resulting energy harvesting WSN, is able to harvest energy and recharge its battery from its environment. Assuming no other failures, WSNs are now able to operate perpectually if all nodes have energy neutral operation; i.e., they harvest more energy than what they spend.

A fundamental problem in energy harvesting WSNs is to maximize coverage, whereby the goal is to capture events of interest that occur in one or more target areas. In this respect, duty cycling for complete targets coverage is important. Its objective is to ensure all targets are monitored continuously by at least one sensor node, whilst other sensor nodes can be in a low power sleep state to conserve energy and to recharge their battery. To date, existing works only study the duty cycling for complete targets coverage problem in non-rechargeable WSNs where the sensor nodes have no ability to refresh their battery. Moreover, past works on energy harvesting WSNs focus on maximizing event detection probability and do not require all targets to be monitored at all times.

To this end, this thesis first addresses the duty cycling for complete targets cov-

erage problem in energy harvesting WSNs. Specifically, it is the first to propose the Maximum Lifetime Coverage with Energy Harvesting node (MLCEH) problem. Its objective is to determine the activation schedule of sensor nodes such that network lifetime is maximal whilst ensuring all targets are monitored by at least one sensor node at all times. To address the MLCEH problem, this thesis proposes two novel solutions, called LP-MLCEH and Maximum Utility Algorithm (MUA). The first uses Linear Programming (LP), and the other relies on a greedy selection policy. Simulation results show that LP-MLCEH doubles the network lifetime obtained by a similar algorithm developed for *finite* battery WSNs. Moreover, MUA achieves 3/4 of the network lifetime achieved by LP-MLCEH at a fraction of LP-MLCEH's computation time.

This thesis also considers a scenario where the information from sensor nodes may be staled, meaning the sink does not have an accurate knowledge of the battery level at each sensor node. To cope with this uncertainty, this thesis proposes a two stage Stochastic Programming (SP) based Uncertain Maximum Lifetime Coverage solution, called SP-UMLC, which achieves 80% of the theoretically achievable network lifetime.

After that, this thesis outlines a distributed algorithm called Maximum Energy Protection (MEP) to allow a sensor node to turn itself on/off using two-hops information. It compares MEP with two distributed algorithms developed for finite battery WSNs. Results show that MEP increases network lifetime by at least 30% and yields lower coverage redundancy.

This thesis then considers connectivity in the context of the Maximum Lifetime Coverage and Connectivity with Energy Harvesting nodes (MLCCEH) problem. The objective is to ensure all activated sensor nodes have at least one path to forward data to the base station/sink. It proposes two solutions: LP-MLCCEH and Energy Conservation for MLCCEH(EC-MLCCEH). Simulation results show that EC-MLCCEH is able to quickly compute results that is within 80% of the network lifetime acquired by LP-MLCCEH.

A key gap in the current state-of-the-art pertaining to coverage using energy harvesting WSNs is that they assume sensor nodes are already deployed, and thus, they do not guarantee a feasible number of sensor nodes around each target to satisfy energy neutral operation. To this end, this thesis studies the nodes placement problem. It first proposes three approximation algorithms, i.e., Greedy Round Node Placement (GRNP), Target Protection Node Placement (TPNP) and Energy Efficient Node Placement (EENP), to determine the locations to place the minimal number of sensor nodes such that the resulting network has energy neutral operation. The proposed algorithms have an approximation ratio of $L+1$, $|Z|+1$ and $\frac{1}{2}|Z|+\frac{3}{2}$, respectively. Here, L is the number of cells and $|Z|$ is the number of targets. In addition, they have a run time complexity of $\mathcal{O}(L)$, $\mathcal{O}(L+\gamma|Z|+2|Z|^2)$ and $\mathcal{O}((L+\lceil\frac{E^c}{R_{min}}\rceil(|Z|+L))$, respectively; The symbol γ and E^c are, respectively, the sensing range and energy consumption rate of sensor nodes, and R_{min} is the minimum recharging rate of cells in the sensing field. Simulation results show that EENP is close to optimal with less than 1% gap in terms of the required number of sensor nodes.

Lastly, this thesis studies the problem of determining the set of locations to place the minimal number of sensor nodes for *both* sensing and relaying. The key constraints include complete targets coverage, energy neutral operation and connectivity to a sink. The problem is formulated as a Mixed Integer Linear Programming (MILP). A challenging issue with the formulated MILP is that it requires all the combinations of locations to place sensor nodes, which is computationally intractable. Consequently, a greedy heuristic called GMILP is proposed to reduce the number of decision variables considered by the MILP. This thesis further proposes two heuristics, i.e., DirectSearch and GreedySearch. Simulation results show that DirectSearch needs to place 20% more sensor nodes than MILP. On the other hand, both GreedySearch and GMILP require 10% more sensor nodes as compared to MILP. Moreover, the running time of DirectSearch and GreedySearch is an order faster than GMILP and four orders faster than MILP.

Acknowledgments

First of all, I would express my deepest gratitude to my supervisor, Associate Professor Kwan-Wu Chin, for his patient guidance, constructive suggestions and constant encouragement throughout my PhD studies. He has set a good example for me in both research and life.

Special thanks goes to my research mates, Mr He Wang, Miss Luyao Wang and Mr Tengjiao He, who are now studying PhD at the University of Wollongong. I wish them all the best in their studies.

Finally, I wish to express my sincere gratitude to my family for their great support and love throughout my life. Without their help, it is impossible for me to reach this stage.

Contents

List of Figures

List of Tables

Abbreviations

BIP	Binary Integer Programming
CPs	Candidate Point(s)
CPNS	Coverage Preserving Node Schedule
DEEPS	Deterministic Energy-Efficient Protocol for Sensing
DMLCEH	Distributed Maximum Lifetime Coverage with Energy Harvesting nodes
DSC	Disjoint Set Covers
EC	Energy Conservation
ECNP	Efficient Coverage Node Placement
EENP	Energy Efficient Node Placement
EH	Energy Harvesting node
ENCC	Energy Neutral Coverage and Connectivity
GMILP	Greedy Mixed Integer Linear Programming
GRNP	Greedy Round Node Placement
i.i.d	independent identical distributed
ILP	Integer Linear Programming
LP	Linear Programming
MAC	Medium Access Control
MCNP	Maximum Coverage Node Placement

MDP	Markov Decision Process
MDSC	Maximal Disjoint Set Cover
MEF	Maximum Energy First
MEHNP	Minimum Energy Harvesting Node Placement
MEP	Maximum Energy Protection algorithm
MILP	Mixed Integer Linear Programming
MLC	Maximum Lifetime Coverage
MLCC	Maximum Lifetime Coverage and Connectivity
MLCCEH	Maximum Lifetime Coverage and Connectivity with Energy Harvesting nodes
MLCEH	Maximum Lifetime Coverage with Energy Harvesting nodes
MSC	Maximum Set Cover
MTF	Maximum Target First
MUA	Maximum Utility Algorithm
MWSC	Minimum Weight Sensor Cover
NP	Nondeterministic Polynomial time
PC	Perpetual Coverage
PDF	Probability Density Function
PEAS	Probing Environment and Adaptive Sleeping
POMDP	Partially Observable Markov Decision Process
RA	Random Activation
RFID	Radio Frequency Identification
SAA	Sample Average Approximation
SP	Stochastic Programming
TPNP	Target Protection Node Placement
UMLC	Uncertain Maximum Lifetime Coverage
WSNs	Wireless Sensor Networks

Chapter 1

Introduction

1.1 Background

A Wireless Sensor Network (WSN) is comprised of a number of inter-connected sensor nodes. In general, a sensor node can measure and collect information from its surrounding. The gathered/sensed data from each sensor node is then transmitted wirelessly to one or more sinks. In turn, the data is then transmitted by the sink to a user for processing. Figure 1.1 shows an example WSN. Each sensor node can monitor its surrounding and routes data to the gateway via other sensor nodes through its wireless transceiver. This means that sensor nodes are able to self-organize and self-configure, and consequently, form a connected network that can be used to monitor and even act on the environment [5].

WSNs have two data transmission models: *direct* and *ad hoc*. In direct transmission mode, sensor nodes are able to connect to a base station/sink directly. On the other hand, in ad hoc transmission mode, sensor nodes have a restricted and usually short transmission range. Thus, a sensor node monitoring targets may also be responsible for forwarding data from other sensor nodes to the base station/sink. For example, as shown in Figure 1.1, sensor nodes that are far away from the sink and gateway require other sensor nodes to help them forward sensed data.

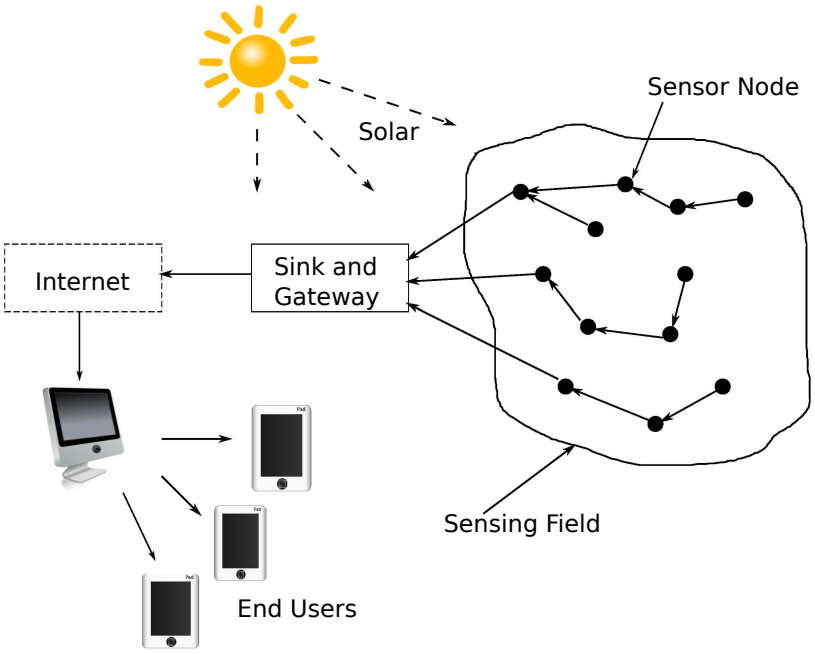


Figure 1.1: A wireless sensor network

A fundamental problem in conventional WSNs is that nodes have a finite battery lifetime. To this end, several researchers have proposed solutions to maximize the lifetime of wireless sensor nodes. Some of these methods include energy-aware Medium Access Control (MAC) protocols, power aware storage, routing, data dissemination protocols, duty-cycling strategies, adaptive sensing rate, tiered system architectures, and redundant placement of nodes [6][7][8][9][10][11]. However, the lifetime of a WSN remains bounded by the battery capacity of sensor nodes.

Table 1.1 lists some WSN applications. Notable ones include monitoring wild life [12], natural resources conservation [13], emergency response [14] and homeland security [15] to name a few. The main task of these applications is to cover a region of interest and to monitor the activities of subjects. We can see that most of these applications are deployed in a remote and hostile environment; e.g., volcano [16], underwater [17] and wilderness areas [13][14][12]. Also, it is worth noting that only the nodes in ZebraNet [12] have solar energy harvesting capability. However, the sensor nodes in [16], [17] and [13] use non-rechargeable batteries and thus they have to be replaced manually. This, however, is not practical. To this end, using energy harvesting technologies to prolong a WSN's operation time is important.

Applications	Key Function	Platform	Sensor Type	Power Generator
Volcanic monitoring [16]	Monitoring volcanic activities	• Texas Instrument MSP430 • 48 Kb with 1 Mb external flash memory • 2.4 GHz Chipcon CC2420 radio using 802.15.4	14 sensor nodes with Geospace Gs-11 Geophone at 4.5 MHz and 2 sensor node at 1 MHz	No
Underwater monitor [17]	Monitoring underwater environment	• 8-bit ATmega128 processor • 128 Kb of program and 512 Kb for data storage • Optical and Acoustic communication module	Pressure, temperature and camera	No
Macroscope of redwood [13]	Monitoring the height of redwood trees	• Atmel ATmega128 processor at 4 MHz • 512 Kb memory • Chipcon 433 MHz radio rate at 40 Kbps	Temperature, humidity and illumination	No
Cenwits [14]	Find missing hiker	• 900 MHz processor • 4 Kb RAM and 128 Kb flash memory • SiRG serial communication protocol	MTS430CA GPS module	No
Pinptr [15]	Voice identity and orientation	• ATmega128 processor at 7.3 MHz • 4 Kb RAM and 128 Kb flash memory • Chipcon CC1000 433 MHz transceiver with 38.4 Kbps	Acoustic sensor operating on a Xilinx Spartan2 FPGA chip	No
ZebraNet [12]	Tracking wild life behaviors	• Texas Instrument MSP430F149 • 2 Kb RAM and 60 Kb internal flash memory • MaxStream 9XStream 900 MHz radio range up to 5 miles	GPS-MS1E GPS chip	Solar

Table 1.1: Some WSN applications

Energy harvesting technologies convert ambient energy to electrical energy. The harvested energy can then be used to drive the load of a sensor node, and if used wisely, a WSN can remain operational for a significant period of time [11]. Table 1.2 shows some examples of energy harvesting sources and their utilization rate. In general, sunlight is the most readily available energy source. Hence, energy harvesting WSNs that are deployed outdoors typically employ solar energy [9, 18, 19]. Current solar technologies have an acceptable conversion efficiency of 15% [11]. In particular, the amount of energy generated by a solar panel is directly proportional to its size and intensity of the incident light. Besides environmental energy, another source of energy is human motion. Interestingly, the resulting energy is sufficient to detect a push of a button [20], and identify respiratory difficulty and Dysarteriotony [3].

Figure 1.2 shows the architecture of an energy harvesting sensor node. It contains the following units:

- *Sensors*: Each unit usually contains one or more sensors with A/D convertors such as temperature, luminosity, and humidity [13] [17]. In WSNs that are used for object tracking, sensor nodes have a GPS module [12, 14].

- *Processor*: Example processors include Atmel ATmega128 [4, 13, 21], ARM7TDMI processor [22] and Texas Instrument MSP430 [16] to name a few. These processors run at a rate of 4 to 12 MHz [13, 22].

- *Memory*: This unit can be divided into RAM, data storage and external storage. In general, sensor nodes can have 4 to 64 KB of RAM and 64 to 512 KB of data storage. Further, a sensor node may have access to an external storage [16].

- *Transceiver*: Example communication methods include radio [16], acoustic, light [4] and Bluetooth [22]. Common transceivers, e.g., Chipcon [13, 15, 16], have a transmission rate ranging from 40 to 750 Kb/s.

Energy Source	**Available Energy**	**Harvesting Technology**	**Conversion Rate (%)**	**Amount of Energy**
Solar	100 mW/cm^2	Solar Cells	15	10 mW/cm^2
Wind	-	Anemometer	-	1200 mWh/day
Finger Motion	19 mW	Piezoelectric	11	2.1 mW
Footfalls	67 mW	Piezoelectric	7.5	5 mW
Indoor Vibrations	-	Electromagnetic Induction	-	0.2 mW/cm^2
Exhalation	1 mW	Breathing masks	40	0.4 mW
Breathing	0.83 mW	Ratchet-Flywheel	50	0.42 mW
Blood Pressure	0.93 mW	Micro-generator	40	0.47 mW

Table 1.2: A comparison of different harvestable energy sources [1][2][3]

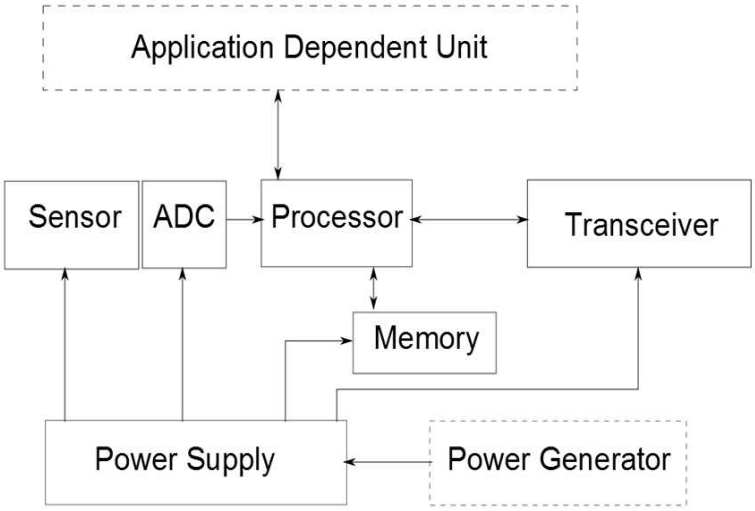


Figure 1.2: Sensor node architecture

- *Power Supply*: Rechargeable battery is the most popular energy source in current WSNs. Lithium based batteries have the highest energy density and charge-discharge efficiency at low self-discharge rate. A research at the University of California showed that super-capacitor, in theory, has unlimited recharge cycles [23], meaning it has the longest lifetime amongst all battery types. However, the weight to energy density ratio of super-capacitors is a significant drawback. A comparison of different batteries is shown in Table 1.3.

- *Power Generator*: This unit is used to harvest ambient energy; i.e., using solar panel, wind turbine, piezoelectric generator or thermoelectric generator [24]. The most accessible energy source is solar energy which can provide 15 mW/cm2. Table 1.2 shows a comparison of different energy sources.

Duty cycling dictates the lifetime of a WSN. In particular, the activation and deactivation schedule of a sensor node controls its energy expenditure and also the time in which it is able to replenish its energy storage. In theory, a WSN with energy harvesting nodes can operate perpetually assuming there is an appropriate duty cycling strategy. This means that the resulting WSN has *energy neutral operation*

Type	**Lead Acid**	**NiCd**	**NiMH**	**Li-ion**	**Li-polymer**	**Supercap**
Model No.	LC-R061R3P	KR-1100AAU	NH15-2500	UBP053048	UBC433475	BCAP0350
Nominal Voltage (V)	6	1.2	1.2	3.7	3.7	2.5
Capacity	1300 mAh	1100 mAh	2500 mAh	740 mAh	930 mAh	350 F
Weight Energy Density (Wh/kg)	26	42	100	165	156	5.06
Charge-discharge Efficiency (%)	70-92	70-90	66	99.9	99.8	97-98
Self Discharge (%/month)	20	10	30	< 10	< 10	5.9/day
Memory Effect	No	Yes	No	No	No	No
Charging Method	Trickle	Trickle/Pulse	Trickle/Pulse	Pulse	Pulse	Trickle
Recharge Cycles	500-800	1500	1000	1200	500-1000	Unlimited

Table 1.3: The different types of rechargeable batteries and their properties [4]

[25]. Specifically, energy neutral operation is achieved when the total harvested energy exceeds a node's energy expenditure incurred due to targets monitoring. For example, consider a target that is covered by five sensor nodes and is required to be continuously monitored by at least one of them. These sensor nodes have a uniform energy harvesting of 10 mW, and they have an energy consumption rate of 50 mW. In this scenario, sensor nodes can take turns to be activated to monitor a target whilst others enter the sleep state to replenish their batteries. Consequently, the sensor nodes can continuously monitor the target for an infinite time. It is worthwhile noting that a sensor node that has a full battery will not be able to harvest any more energy and thus it is required to expend it.

An important design constraints in WSNs is *coverage ratio*. This is the percentage of points of interest or targets within the sensing field that are monitored by sensor nodes. For example, if 70 out of 100 targets in a WSN are covered by at least one sensor node, then the coverage ratio is 70%. A network provides *complete* targets coverage if it ensures 100% coverage ratio. This is a desirable characteristic in many surveillance applications because it means that all events that occur at a target location will be recorded. For example, in the volcanic monitoring application outlined in [16], all shocks must be recorded so that ample warnings can be given to evacuate people before a severe eruption or earthquake. In general, a point of interest or target within a sensing field is usually covered by more than one sensor node. Consequently, having all sensor nodes in the active state to ensure complete targets coverage means that there will be redundant coverage, and more importantly, it wastes precious energy. To this end, duty cycling is critical. It ensures that a redundant sensor node is powered down temporarily whereby the sensor node's key functions such as sensing, transmission and computation, are switched off to conserve energy.

1.2 Problem Space and Motivation

The key observation that initiated this thesis is that existing works on complete targets coverage do not consider sensor nodes with energy harvesting capability. Note, targets are assumed to have a fixed location. Specifically, prior works on coverage aim to maximize network lifetime over nodes with *finite* battery. Henceforth, this thesis aims to fill this critical gap given the importance of energy harvesting WSNs. It considers two key problems: (i) duty cycling sensor nodes such that they provide complete targets coverage and have ample opportunities to recharge, and (ii) placing the minimal number of nodes in a manner that achieves *energy neutral coverage* as well as complete targets coverage.

1.2.1 Duty Cycling

The main aim of duty cycling is to maximize a network's lifetime subject to the following design constraints: complete targets coverage, recharging opportunity, random recharging rate and/or connectivity. In other words, given a set of time synchronized sensor nodes, determine a subset of these sensor nodes to activate whilst placing others to sleep. The activated nodes must cover all targets. Moreover, any developed algorithms must consider the random energy harvesting rates experienced by sensor nodes at different times and locations [26]. For example, when using a solar panel, the portion of the incident solar energy available at the panel is determined by a variety of environmental factors such as clouds and foliage [4]. Besides the above considerations, an additional requirement is to ensure all active sensor nodes form a connected network. Each active sensor node can then forward/relay sensed data to a sink.

As an example, consider Figure 1.3. Eight sensor nodes are used to monitor two targets indicated by a triangle. The sensing and communication range of each sensor node is indicated by a circle around it. The sink, labeled B, is shown as a square. Each sensor node has an independent identical distributed (i.i.d) recharging

rate. In order to ensure coverage and connectivity constraints, one can first activate sensor nodes in the set $\{s_1, s_2, s_3, s_4\}$, where sensor nodes s_1 and s_3 are used for targets coverage whilst s_2 and s_4 are responsible for forwarding sensed data to B. When the energy level of sensor nodes in $\{s_1, s_2, s_3, s_4\}$ is low, another set of sensor nodes, say $\{s_5, s_6, s_7, s_8\}$, can be activated. Thus, the sensor nodes in the set $\{s_1, s_2, s_3, s_4\}$ can enter sleep mode; i.e., a low power state. In order to maximize recharging opportunities, sensor nodes in the set $\{s_1, s_2, s_3, s_4\}$ should be activated before their batteries are at capacity. Moreover, the activation schedule should allow for sensor nodes to recharge and avoid any missed recharging opportunities due to a full battery. The latter point means sleeping sensor nodes with a full battery must be activated to allow for future recharging opportunities.

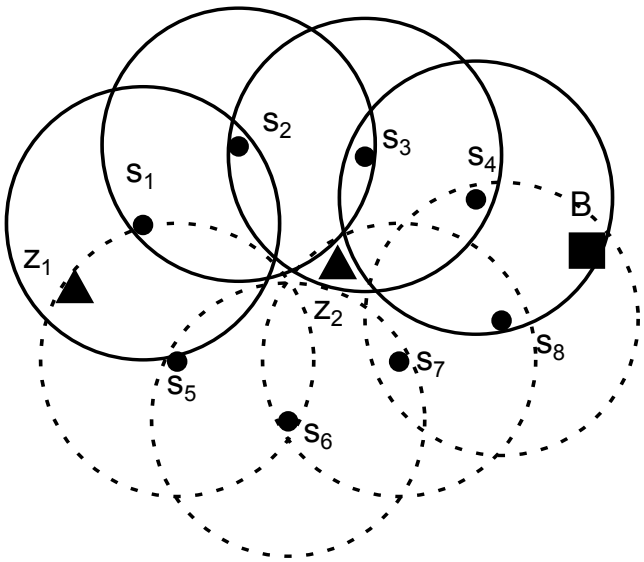


Figure 1.3: An example network.

A key challenge in the foregone example is to ensure that the base station or sink, which is responsible for controlling the duty cycle of sensor nodes, knows the exact battery level of each sensor node. This information is required because the schedule computed by the base station ensures that sensor nodes remain awake to monitor a target or remain in sleep mode to recharge and ensures they expend stored energy. Otherwise, due to random recharging rates, upon receiving an activation schedule from the sink, a sensor node may have insufficient energy to implement

the schedule. Conversely, a sensor node may experience a temporary but "high" recharging rate that allows it to recharge fully. For example, sensor nodes s_1 and s_5 can only receive their activation schedule after at least two hop transmissions. During this period, they may experience a change in recharging rate and thus making the schedule, which was computed based a different battery level, invalid. One way to reduce staled battery level information at the sink is to require sensor nodes to send frequent updates to the sink. This, however, is at the expense of precious energy.

1.2.2 Nodes Placement

Past works, see [27], that consider duty cycling assume that all sensor nodes are already deployed randomly around targets; for example, by dropping them from a passing plane. This, however, does not guarantee that there are sufficient number of sensor nodes around each target to ensure energy neutral operation. Moreover, redundant sensor nodes increase deployment cost. Another key consideration is that in practice the sensing field has a different *potential recharging rate* at each location. This is the amount of harvested energy if a sensor node is placed on it.

A key problem is thus to determine good locations to place nodes and their numbers in order to minimize the total number of deployed sensor nodes. The main constraints include energy neutral operation and/or coverage, random recharging rates and/or connectivity. Consider Figure 1.4, which shows three cells A, B and C that can be used to place sensor nodes to monitor two targets denoted by a 'star'. The power required to monitor the targets is 100 milliwatts. If a sensor node is placed in cell A, then it is able to monitor both targets. However, if sensor nodes are placed in cell B or C, they can only monitor one target. As indicated, the potential energy harvesting rate of these cells is respectively 20, 100 and 100 milliwatts. Thus, energy neutral coverage is achieved by either placing five sensor nodes in cell A, or by placing one sensor node in both cell B and C. It can be seen

that the key challenge is to determine cells that have both a high energy harvesting rate and are able to cover many targets. If such cells exist, then the number of required sensor nodes is reduced. This is because all targets can be monitored from a small number of cells, and advantageously, each cell only needs to have a few sensor nodes in order to achieve energy neutral coverage. Moreover, the location to place relaying nodes to forward sensed data to the base station/sink is also important given that sensor nodes require multi-hop communications to forward sensed data back to the sink.

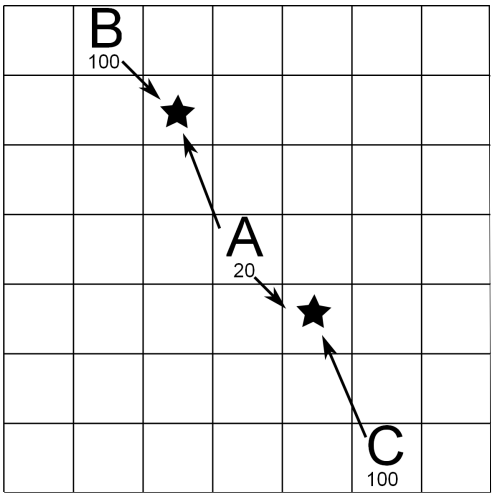


Figure 1.4: An example of the node placement problem. 'Stars' are static targets. The numbers indicate the potential recharging rate of a cell. Each location may have multiple sensor nodes.

1.3 Contributions

This thesis aims to address the aforementioned problems. The main contributions, in terms of developed algorithms that address these problems, are listed in Table 1.4. The details of the said contributions are as follows.

1.3.1 Maximum Lifetime Coverage with Energy Harvesting node (MLCEH)

The goal is to schedule the active/sleep state of sensor nodes to maximize an energy harvsting WSN's lifetime whilst ensuring complete targets coverage. This thesis first proposes a Linear Programming (LP) based LP-MLCEH solution subject to coverage and energy constraints. The simulation results show that LP-MLCEH achieves more than twice the performance in terms of network lifetime as compared to similar algorithms developed for finite battery WSNs. However, it is computationally expensive. To address this limitation, the Maximum Utility Algorithm (MUA) is proposed, which achieves 3/4 of the network lifetime obtained by LP-MLCEH using only a fraction of the computation time of LP-MLCEH.

1.3.2 MLCEH with Random Recharging Rates

As mentioned, the sink may not have an up to date information regarding the battery level of nodes. This is due to the random recharging rates experienced by sensor nodes and also delays in propagating battery level information over multiple hops to the sink. To this end, the thesis presents a Stochastic Programming based Uncertain Maximum Lifetime Coverage (SP-UMLC) approach. It considers sensor nodes with energy harvesting capability and also have a non-rechargeable battery which they use to meet the shortfall in energy required to remain active. This approach solves the MLCEH problem with random recharging rates problem via a two-stage Stochastic Program (SP) with the goal of minimizing the expected recourse cost, i.e., the amount of energy a sensor node draws from its battery reserve in order to meet any shortfall in energy. A key challenge in using a SP approach is the exponential number of scenarios, where each scenario is a realization of the battery level of all sensor nodes. Consequently, as the number of battery levels or nodes increases, the resulting SP formulation becomes intractable. To this end, the thesis employs the Sample Average Approximation (SAA) framework to solve the formulated two-stage

SP [28]. Experimental results show the proposed SP approach achieves 80% of the theoretically achievable coverage lifetime.

1.3.3 Distributed MLCEH

The aforementioned centralized algorithms, i.e., LP-MLCEH, MUA and SP-UMLC, require a centralized controller, which is responsible for gathering information, and deriving and disseminating an active/sleep schedule to sensor nodes. This, however, incurs a high energy expenditure due to communications. Henceforth, this thesis considers a distributed version of the MLCEH problem, aka DMLCEH. Specifically, the aim is to allow a sensor node to choose its active/sleep state based on its information and that of its one hop neighbours. To this end, the thesis outlines the Maximum Energy Protection (MEP) algorithm. The main idea is to replace on-duty sensor nodes with those currently in sleep state that have higher energy level. It also ensures sensor nodes do not lose any recharging opportunities. Moreover, it minimizes redundant coverage whereby a target is covered by sensor nodes that can be switched off whilst ensuring all targets are monitored by at least one sensor node. Simulation results show MEP increases network lifetime by 30%, and has lower coverage redundancy as compared to two similar distributed algorithms developed for finite battery WSNs.

1.3.4 Maximum Lifetime Coverage and Connectivity with Energy Harvesting node (MLCCEH)

A key assumption of the MLCEH sub-problem is that sensor nodes are able to connect to the base station/sink directly. To relax this assumption, this thesis considers the MLCEH problem with connectivity constraint; aka the MLCCEH problem. The objective is to design a centralized algorithm to schedule the active/sleep time of sensor nodes such that network lifetime is maximum whilst ensuring complete targets coverage and connectivity. To solve the MLCCEH problem, this thesis proposes a

Linear Programming based MLCCEH (LP-MLCCEH) approach subject to coverage, energy and flow conservation constraints. The constraints correspond to complete targets coverage, a sensor node only uses its available energy and all nodes have connectivity to the sink. In addition, thesis also contains an efficient heuristic algorithm for MLCCEH because the LP solution requires an exhaustive collection of set covers. The heuristic, called Energy Conservation for MLCCEH (EC-MLCCEH), iteratively selects sensor nodes with a high residual energy to monitor targets or to forward sensed data until the resulting set of sensor nodes provide the required coverage and connectivity to the sink. The simulation results show that EC-MLCCEH achieves 80% of the network lifetime computed by the LP-MLCCEH at a fraction of the computation time.

1.3.5 Minimum Energy Harvesting Node Placement for Perpetual Coverage (MEHNP-PC)

This thesis then considers the nodes placement problem. Specifically, it addresses a sub-problem which aims to design a centralized algorithm to place the minimum number of sensor nodes such that the total harvested energy exceeds what is required to monitor all targets; aka the MEHNP-PC problem. The main issue is to determine the locations to place sensor nodes, which have a different coverage and more importantly, each location has a different recharging rate and thus will need varying number of sensor nodes to be placed in order to achieve energy neutral operation/coverage. To address the said problem, this thesis proposes three approximation algorithms; namely GRNP, TPNP and EENP. It proves their approximation ratio to be $L+1$, $|Z|+1$ and $\frac{1}{2}|Z|+\frac{3}{2}$, respectively. Here, L is the number of cells and $|Z|$ is the number of targets. In addition, they have a run time complexity of $\mathcal{O}(L)$, $\mathcal{O}(L+\gamma|Z|+2|Z|^2)$ and $\mathcal{O}((L+\lceil\frac{E^c}{R_{min}}\rceil(|Z|+L))$, respectively; γ and E^c are the sensing range and energy consumption rate of sensor nodes, and R_{min} is the minimum recharging rate of cells in the sensing field. Simulation results show that

EENP is close to optimal with less than 1% gap in terms of the required number of sensor nodes.

1.3.6 Minimum Energy Harvesting Node Placement for Energy Neutral Coverage and Connectivity (MEHNP-ENCC)

A key limitation of MEHNP-PC problem is that it assumes sensor nodes are able to communicate with the base station/sink directly. In other words, the MEHNP-PC problem does not consider the network connectivity constraint. To this end, this thesis addresses a sub-problem, called MEHNP-ENCC. The main difference to the MEHNP-PC problem is that it aims to determine the minimum number of sensor nodes respectively for sensing and relaying, and their locations. The design constraints include energy neutral coverage, random recharging rates and connectivity. To address this sub-problem, this thesis first proposes a Greedy Mixed Integer Linear Programming (GMILP) algorithm. This algorithm uses a greedy heuristic to determine a collection of set covers of locations that ensure complete targets coverage. It then uses a MILP to determine the number of sensor nodes in these set covers to ensure network connectivity. However, the MILP incurs a high computational cost. To this end, this thesis then proposes two heuristics, namely DirectSearch and GreedySearch, to iteratively determine the locations to place sensor nodes for sensing and relaying. Simulation results show that DirectSearch requires 20% more sensor nodes than the optimal solution whilst this value is 10% for GreedySearch and GMILP. However, the running time of DirectSearch and GreedySearch is an order faster than GMILP.

1.4 Publications

The work in this thesis has resulted in the following papers:

Problems	**Algorithms**	**Transmission Model**	**Algorithm Type**	**Comments**
MLCEH	LP-MLCEH	Direct	LP	
	MUA	Direct	Heuristic	
	SP-UMLC	Direct	SP	Uncertain battery levels
	MEP	Direct	Heuristic	Distributed algorithm
MLCCEH	LP-MLCCEH	Ad hoc	LP	
	EC-MLCCEH	Ad hoc	Heuristic	
MEHNP-PC	EENP, TPNP and GRNP	Direct	Approximation	
MEHNP-ENCC	GMILP	Ad hoc	MILP and Heuristic	Coverage and connectivity
	DirectSearch and GreedySearch	Ad hoc	Heuristic	

Table 1.4: A list of contributions

1. **C.L. Yang** and K-W Chin. *Novel algorithms for complete targets coverage in energy harvesting wireless sensor networks*, IEEE Communications Letters, 18(1), p118-122, January, 2014.

2. **C.L. Yang** and K-W Chin. *On complete targets coverage in wireless sensor networks with random recharging rates*, IEEE Wireless Communications Letters, 4(1), p50-53, 2015.

3. **C.L. Yang** and K-W Chin. *A novel distributed algorithm for complete targets coverage in energy harvesting wireless sensor networks*, IEEE International Conference on Communications (ICC), Sydney, Australia, June, 2014.

4. **C.L. Yang** and K-W Chin. *On Complete Targets Coverage and Connectivity in Energy Harvesting Wireless Sensor Networks*, IEEE 22nd International Conference on Telecommunications (ICT), Sydney, Australia, April, 2015.

5. **C.L. Yang** and K-W Chin. *On Nodes Placement for Energy Neutral Targets Coverage in Energy Harvesting Wireless Sensor Networks*, IEEE Transaction on Industrial Informatics. *Under second review*

6. **C.L Yang** and K-W Chin. *On Minimum Node Placement for Energy Neutral Complete Targets Coverage and Connectivity in Energy Harvesting WSN*, IEEE Transaction on Industrial Informatics. *Under Review*

1.5 Thesis Structure

1. *Chapter 2.* This chapter includes a literature review of existing duty cycle scheduling and nodes placement approaches on the coverage problem in both conventional non-rechargeable and energy harvesting WSNs.

2. *Chapter 3.* This chapter introduces the MLCEH problem and outlines the LP-MLCEH and MUA algorithm.

3. *Chapter 4.* This chapter proposes the SP-UMLC algorithm to cope with the uncertain battery levels at the sink.

4. *Chapter 5.* This chapter presents the MEP algorithm to address the DMLCEH problem.

5. *Chapter 6.* This chapter outlines the MLCCEH problem and proposes the LP-MLCCEH and EC-MLCCEH algorithms.

6. *Chapter 7.* This chapter proposes and analyse three approximation algorithms, i.e., EENP, TPNP and GRNP, for the MEHNP-PC problem.

7. *Chapter 8.* This chapter proposes the GMILP, DirectSearch and GreedySearch algorithms to address the MEHNP-ENCC problem.

8. *Chapter 9.* This chapter concludes the thesis, and provides a summary of research outcomes and future research directions.

Chapter 2

Literature Review

This chapter reviews prior works on duty cycling and nodes placement. The focus will be on two types of WSNs: non-rechargeable and energy harvesting. A summary is then presented in Section 2.3.

2.1 Duty Cycling Algorithms

This section first considers the duty cycling problem for WSNs where sensor nodes are densely deployed and are non-rechargeable. Specifically, Section 2.1.1 introduces both centralized and distributed solutions for the Maximum Lifetime Coverage (MLC) problem, which aim to prolong network lifetime whilst ensuring all targets are covered by at least one sensor node. Algorithms/solutions reviewed also include those that solve the MLC problem and ensuring connectivity to a base station; aka the MLCC problem; see Section 2.1.2. Then Section 2.1.3 covers the duty cycling problem in energy harvesting WSNs.

2.1.1 Maximizing Lifetime Coverage

The MLC problem calls for a solution to prolong the network lifetime of a WSN such that all targets are covered by at least one sensor node. Lifetime is defined as the

time duration from the time instant when a WSN starts operation to the time when a target is not watched by any sensor nodes. In a nutshell, the MLC problem is as follows: given a set of sensor nodes and a set of targets, find a schedule that activates these sensor nodes in a manner that monitors the targets for maximum time. Note that in this problem all sensor nodes are assumed to have a direct communication with a central controller.

To address the MLC problem, a common approach requires each sensor node to forward all its information such as location, target(s) covered and residual energy to a central controller. This central controller can be a sensor node that has ample energy supply and computational capability. The central controller then calculates the activation time for all sensor nodes. A key constraint is that there must be minimal redundant coverage. This ensures that the energy of sensor nodes is used efficiently in a manner that prolongs network lifetime. To this end, the most common activation solution is to organize sensor nodes into various subsets [29–36]. Each subset is a set cover that consists of sensor nodes that monitor all targets completely. Hence, when a set cover is active, all targets are covered. Moreover, the sensor nodes in other set covers can enter the sleep state to conserve their energy.

An optimal activation schedule for each set cover can be derived via Linear Programming (LP) [37]. The objective is to maximize network lifetime subject to complete targets coverage and finite battery capacity constraints. A key assumption is that all sensors are initialized with one unit of energy, and have the same energy consumption rate when active. In other words, if a set cover is active continuously, then all sensors in the set cover will expend energy for one time unit. Mathematically, given a collection of set covers $\{C_1, C_2, \ldots, C_j, \ldots, C_J\}$ and sensor nodes $\{s_1, s_2, \ldots, s_i, \ldots, s_I\}$, let $\{t_1, t_2, \ldots, t_j, \ldots, t_J\}$ indicate the active interval of each cover, and $s_{ij} \in \{0, 1\}$ indicate whether sensor s_i is in the set cover C_j. Moreover, let $Z(C_j) = 1$ indicate all targets monitored by cover C_j. The LP for the MLC problem is as follows,

$$\text{MAX} \quad \sum_{j=1}^{J} t_j \tag{2.1}$$

$$\text{s.t.} \quad \sum_{j=1}^{J} s_{ij} t_j \leq 1, \quad \forall s_i \tag{2.2}$$

$$Z(C_j) = 1, \quad \forall C_j \tag{2.3}$$

$$s_{ij} = \begin{cases} 1 & s_i \in C_j \\ 0 & s_1 \notin C_j \end{cases} \tag{2.4}$$

The objective of this LP is to maximize the sum of lifetime of each set cover; i.e., $\sum_{j=1}^{J} t_j$. Constraint 2.2 ensures that the active duration of each sensor node is limited by its battery lifetime; i.e., one unit time. Constraint 2.3 ensures that all targets are monitored; i.e., $Z(C_j) = 1$. In this LP, the fundamental problem is how to construct the collection of set covers; i.e., $\{C_1, C_2, \ldots, C_j, \ldots, C_J\}$. To do this, current approaches either divide sensors into disjoint set covers [33, 35, 36] or allow sensor nodes to participate in multiple set covers [29–32, 34].

Disjoint Set Covers

This subsection introduces solutions for the MLC problem that involve partitioning sensors into disjoint set covers. Note that two set covers are called disjoint if their intersection is an empty set. In other words, a sensor node can only be in at most one set cover. The resulting disjoint set covers are then activated one after another. This means only sensor nodes in the active set cover are activated while other sensor nodes remain in the low power sleep state. The sensor nodes in the active set cover continue to monitor target(s) until their battery runs out. Consequently, the total network lifetime is equal to the number of disjoint set covers multiplied by the active duration of each set cover. Therefore, determining the maximum number of disjoint set covers has a direct impact on network lifetime. However, determining

the optimal collection of disjoint set covers is a NP-complete problem [33].

Slijepcevic et al. [36] first propose a heuristic solution with an objective to maximize the network coverage lifetime. The basic idea is to minimize coverage redundancy; i.e., multiple sensor nodes that cover the same target(s). They construct a set cover by iteratively selecting a sensor node such that there is minimal redundancy with respect to existing sensor nodes already in the given cover. This continues until all targets are monitored.

Cardei et al. [33] propose the Maximal Disjoint Set Cover (MDSC) problem to find the maximum number of disjoint covers. They transform MDSC into a maximum-flow problem containing two sinks. The number of flows to the first sink represents the number of disjoint set covers. They use a mixed integer program to maximize the flow to the first sink and calculate the flow/capacity of each link. The key constraint is that the flow of each link that is connected directly to the first sink can only be full or zero. This represents a sensor node that can only be in at most one set cover. Moreover, the flow/capacity on each link indicates whether a sensor node is in a cover.

Ahn et al. [35] propose a Binary Integer Programing (BIP) formulation to find the maximum number of disjoint set covers. This BIP formulation has two constraints: a sensor node can be included in at most one set cover, and sensor nodes in each set cover are able to monitor all targets. However, if the size of the network becomes large, this formulation becomes intractable. To this end, they propose a heuristic based on the relaxed BIP to compute such disjoint set covers. They first set a possible maximum number of set covers and run the relaxed BIP. They then round up the fractional results of the relaxed BIP to one. If the number of decision variables that are set to one equals the maximal number of set covers, the heuristic stops. This means the resulting number of set covers is maximum. Otherwise, the heuristic reduces the size of the set covers by one and it runs the relaxed BIP again. This process continues until the heuristic obtains the desired number of set covers.

Scheduling sensor nodes into disjoint set covers significantly increases network

lifetime because it minimizes redundant sensor nodes being used to monitor targets. However, allowing a sensor node to be a member of multiple set covers leads to a better solution. For example, consider a set of sensor nodes $\{s_1, s_2, s_3\}$ and targets $\{z_1, z_2, z_3\}$ with a sensor-target topology s_1-z_1-s_2-z_2-s_3-z_3-s_1. Each sensor node can be in the active state for one unit time. If we organize these sensor nodes into disjoint set covers, at most one set cover can exist; i.e., $\{s_1, s_2\}$, $\{s_1, s_3\}$ or $\{s_2, s_3\}$. One of these set covers will be activated continuously for one unit time. This yields a system lifetime of one unit time because the remaining sensor nodes with available energy is not able to cover all the targets. On the other hand, assume the sensor nodes can be activated in multiple set covers; i.e., the non-disjoint set covers case. We can activate the set covers as follows: $\{s_1, s_2\}$ for 0.5 unit time, followed by $\{s_2, s_3\}$ for 0.5 unit time, then $\{s_1, s_3\}$ for 0.5 unit time. In this case, we have a total system lifetime of 1.5 unit time. This yields a network lifetime gain of 0.5 unit time over the disjoint set covers case.

Non-disjoint Set Covers

This subsection considers the scenario where set covers need not be disjoint. That is, a sensor node is allowed to participate in multiple set covers. Each so called non-disjoint set cover is activated in a scheduled sequence to maximize system lifetime. Past work [30] shows that the network lifetime can be extended by using maximal non-disjoint set covers. Therefore, a common sub-problem is to find the maximum number of non-disjoint set covers.

Cardei et al. [30] model the maximum lifetime coverage problem as a Maximum Set Cover (MSC) problem, which is to find the maximal non-disjoint set covers. They solve the MSC problem using an integer program with the objective to maximize the cumulative lifetime of set covers. The constraints include energy and coverage, and whether a sensor is in a cover. They then relax integral constraints. After which, they prove that the MSC problem is NP-complete and propose a greedy heuristic solution. The proposed heuristic iteratively builds a set cover by assigning

a sensor node with the greatest contribution to monitor each target until all targets are covered. For example, a sensor has a greater contribution if it covers a large number of uncovered targets or if it has more residual energy.

Berman et al. [34] propose a Minimum Weight Sensor Cover (MWSC) problem that aims to construct non-disjoint set covers with minimal number of sensor nodes. They first formulate the problem as a packing LP [34] with the objective of maximizing the activation time of each set cover. The key constraint is that the active duration of each cover is limited by a node's battery lifetime. They then use the Garg-Könemann algorithm [38], which is a fast, approximation algorithm for the formulated packing problem, to find the covers with the minimum number of sensor nodes.

Ahn et al. [31] propose a new LP formulation for the MSC problem derived in [30]. They use a set of linear constraints to indicate whether a sensor node is in a set cover. They also propose a LP for the MWSC sub-problem derived in [34] with the objective to minimize the number of sensor nodes in a set cover subject to the constraint that a sensor node must be in at least one cover. However, this LP is computationaly expensive. They therefore propose a heuristic that iteratively constructs a set cover by adding sensor nodes that yield the minimum coverage redundancy. If the heuristic succeeds in generating a set cover that improves the objective function of the LP for the MSC problem, it stops and adds the set cover to the final solution. Otherwise they use the LP for the MWSC problem to find the minimum weighted cover and add the cover to the final solution. They then assign an activation time to the resulting set covers using another LP with the objective of maximizing system lifetime and energy constraint. They initially set the lifetime to the value obtained from a heuristic procedure suggested in [30]. Then, the stabilization column generation technique proposed in [39] is applied to accelerate convergence.

Pyun et al. [32] consider a scenario where the energy consumption rate increases proportionally as per the number of targets watched by a sensor. To solve the

MLC problem, they propose an algorithm to first exhaustively search all possible matchings of sensor nodes and targets. This algorithm then derives the optimal solution from an integer program that is based on the one in [30]. That is, the objective is to maximize the sum of lifetime of each set cover subject to energy and coverage constraints. They also propose a heuristic algorithm that iteratively constructs covers by selecting sensor nodes that cover a large number of uncovered targets.

The authors of [29] use a coverage matrix to represent whether a sensor node can monitor a target. They introduce both an optimal and an approximation algorithm to solve the MLC problem based on this coverage matrix. In the optimal algorithm, they exhaustively search all non-disjoint set covers, and solve the MLC problem using a LP. Specifically, the objective of the LP is to maximize the sum of lifetime of each cover and the key constraint is finite energy. In their approximation algorithm, they iteratively construct a cover by selecting sensor nodes that cover a large number of uncovered targets. If several sensor nodes cover the same number of uncovered targets, they select the one that has a lower redundancy.

The aforementioned algorithms that organize sensor nodes into disjoint or non-disjoint set covers require sensor nodes to forward their location, battery level and sensed data information to a central controller. These so called centralized algorithms thus incur non-negligible communication cost. Moreover, if a sensor node fails, it results in coverage loss. To avoid these drawbacks, another research direction is to employ distributed algorithms, whereby a sensor node decides whether to become active based on the knowledge of its own, and one or two hop neighbors. Compared to centralized algorithms, distributed algorithms reduce energy expenditure due to communications, and hence are more suitable for large-scale WSNs.

A key consideration when designing distributed algorithms is to save the energy of sensor nodes whilst ensuring a given coverage requirement. To this end, a common approach is to activate a sleeping sensor node periodically to communicate with its neighbors to check for redundancy and to turn off overlapping sensor nodes [34, 40].

However, as we will see later, the protocols developed in [41] and [42] equip sensor nodes with a trigger circuit that responds to signal at a specific frequency even in the sleep state. Specifically, this can be achieved using a novel technique suggested in [43] whereby the authors propose a wake-up trigger that consumes no energy.

Tian et al. [40] propose an algorithm that allows as many sensor nodes as possible to be turned off most of the time. They divide time into equal length rounds. Each round begins with a two steps self-scheduling phase followed by a sensing phase. In the self-scheduling phase, each node exchanges its location and the targets it covers with its neighbors. If a sensor node finds that its sensing area is covered fully by its neighbors, it switches off. Otherwise, it activates itself in the sensing phase.

Berman et al. [34] also propose a distributed algorithm whereby each sensor node is in one of three states: active, idle and vulnerable. In the idle state, a sensor node does not monitor its region but listens to other sensor nodes. In the vulnerable state, a sensor node monitors its region but changes its state to either active or idle as soon as possible. A vulnerable sensor node will turn to active if there are targets in its sensing range not covered by other active or vulnerable sensors. It will turn to idle if all its targets are covered by active or vulnerable sensors with a larger energy supply. When a sensor node is in the active or idle state, it goes into the vulnerable state if any neighboring sensor nodes change their state.

Brinza et al. [41] propose a Deterministic Energy-Efficient Protocol for Sensing (DEEPS) implemented from [34]. In this protocol, a sensor node can be in one of three states: working, sleeping and alert. In the alert state, a sensor node needs to switch to the working or sleeping state immediately. Each alert sensor node volunteers to monitor a target that is not covered by working or alert nodes, and goes to the sleeping state if all targets are already covered by working nodes. A working node sends a trigger, see [43], when it runs out of energy to wake-up neighboring nodes. The key difference between [41] and [34] is that sensor nodes in [41] that are in the sleep state do not need to turn on their receiving unit to listen to its neighbors. Details can be found in [43].

He et al. [44] design a large scale wireless sensor network system called VigiNet to track targets. They assume events of interest or targets occur at a relatively low rate; e.g., five minutes per day. Each sensor node may be in one of two states: sentry and non-sentry. In the sentry state, a sensor node monitors its surrounding for a predetermined time period with respect to a predicted target's movement speed. In the non-sentry state, a sensor node periodically wakes up to receive a potential wake-up message: it will turn into the sentry state if (1) it receives a wake-up message, or (2) none of its neighbor is in the sentry state. A sensor node periodically switches between the sentry and non-sentry states if there is no target in its sensing range. Whenever a sentry sensor node detects a target, it will send a wake-up message to all non-sentry nodes within its communication range to cooperatively track the target. These sensor nodes continue to track the target until it moves out of their sensing range.

Islam et al. [42] propose a distributed algorithm in which all sensor nodes have the same sensing range and is aware of their distance to each neighbor. The authors assume each sensor node is able to calculate the percentage of its overlapping area with its neighbors based on their distance and sensing range. This algorithm has three states: sleeping, checking and working. When a sleeping sensor node goes into the checking state, it exchanges its location and battery level information with its working neighbors to calculate the percentage of overlapping area. It will then go to the working state if one of its neighbors' distance is (1) less than the sensing range and the neighbor has 20% lower residual energy, or (2) within 1 to 1.732 sensing range and the percentage of overlap is lower than a predetermined value, or (3) more than 1.732 times the sensing range. Otherwise, a sensor node goes back to the sleeping state. Its sleep duration is proportional to its distance to the working node if they are within each other's sensing range. If not, the percentage of overlapped area becomes a key determinant effecting their sleep time. When a working sensor's energy is below a predetermined threshold, it will wake up an arbitrary sleeping neighbor using the trigger-wakeup technique of [43].

Ye et al. [45] introduce a distributed scheduling protocol, called PEAS, for WSNs operating in harsh environments where sensor nodes may fail frequently. Sensor nodes can be in one of three states: sleeping, probing and working. When a node enters the probing state, it starts sensing its surrounding. If it detects other working nodes, it will calculate its sleep duration according to these working nodes' residual energy and number of sleeping nodes around a given working node. It chooses the shortest sleep time calculated with respect to all of its working neighbors before going back to sleep. On the other hand, a probing node will enter the working state if there is no working node. It continues to be active until failure.

Yan et al. [46] propose a protocol that works in equal length rounds. Each round contains an initialization phase followed by a sensing phase. In the initialization phase, sensor nodes watching the same target will build a schedule so that they are active in turns and their total working time is equal to the duration of the sensing phase. A sensor node will build a schedule for each target in its sensing range. For a sensor node that covers multiple targets, its activation schedule is the sum of the individual schedules of all the targets it covers. In the sensing phase, each sensor node following its own schedule switches states between sleeping and working.

Prasad et al. [47] propose a distributed algorithm where each sensor node can be a central node that constructs a collection of set covers out of its neighbors. Each cover is a group of sensor nodes that monitor all targets within the central node's sensing range. They assume the sensing range is one-half of the communication range. The algorithm begins with an initialization phase followed by a reshuffle phase. In the initialization phase, each sensor node acting as a central node exchanges its location, energy level and targets with its neighbors and constructs a collection of set covers. The set covers do not necessarily include the central node itself. Each central node then prioritizes its calculated covers according to the minimal lifetime of a sensor node. A shorter lifetime means a lower priority. In the reshuffle phase, time is divided into equal length rounds. Each sensor node, including the central node, decides the on/off status of its covers at the beginning of each

round. It starts with the highest priority cover as the most desirable configuration for its neighborhood. If the same sensor nodes are selected by its neighbors, then this cover becomes active. Otherwise, it transitions to the next best priority cover.

Zhang et al. [48] propose a protocol where sensor nodes with a higher residual energy will voluntarily cover a target. The authors divide time into equal length rounds. Each round begins with an initialization and followed by a sensing phase. In the initialization phase, a sensor node first exchanges its location and targets it can cover with its neighbors. Then it waits for a time duration calculated based on its residual lifetime. It should be noted that each sensor node's residual lifetime indicates its activation priority. A sensor node, at the end of its waiting time, will activate itself in the sensing phase if there are uncovered targets. If all targets are covered by its neighbors, it goes to the sleep state.

In summary, the operation of the aforementioned distributed algorithms can be classified into two models: activate a sensor node when necessary [34, 41, 42, 45], and activate sensor nodes after a decision phase [40, 44, 46–48]. An algorithm that belongs to the second model needs to ensure all sensor nodes enter the decision phase at the same time. This can be achieved by broadcasting a time synchronization message to all sensor nodes periodically. If a sensor node fails unexpectedly, it will be replaced by one of its neighbors as soon as they detect the failure. This failure is detected by a wake up neighbor in the first model, or another decision phase in the second model. Moreover, information exchange is local and only occurs between two-hop neighbors, which significantly reduces communication overheads as compared to centralized algorithms.

2.1.2 Maximizing Lifetime Coverage and Connectivity

The Maximizing Lifetime Coverage and Connectivity (MLCC) problem extends the MLC problem. It aims to prolong the time in which all targets are covered by at least one sensor node. Moreover, it considers the need for sensor nodes to send

sensed data to a base station via a multi-hop path. A node may function only as a relay where it receives and forwards sensed data from other nodes. Other nodes may act as a relay and are also responsible for sensing targets. The network lifetime for the MLCC problem is defined as the period from the time when the network is set up to the time the network cannot meet its coverage or connectivity requirement. One common solution is to divide sensor nodes into a family of subsets, each individually maintaining network coverage and connectivity requirement [30, 49–51]. An approach is to formulate the MLCC problem as a LP that is similar in formulation to the MLC problem; see Section 2.1.1. However, a flow conservation constraint is included to ensure nodes in the calculated set covers are connected. Consequently, for each sensor node, the total amount of data sensed and received must equal the amount of data transmitted. Let R denote the data collection rate of each activated sensor node, f_{ik} denotes the amount of data transmitted from sensor node s_i to s_k and the total number of sensor node is I. The MLCC problem can be formulated as follows,

$$\text{MAX} \quad \sum_{j=1}^{J} t_j \tag{2.5}$$

$$\text{s.t.} \quad \sum_{j=1}^{J} s_{ij} t_j \leq 1, \qquad \forall s_i \tag{2.6}$$

$$Z\{C_j\} = 1, \qquad \forall C_j \tag{2.7}$$

$$R \sum_{j=1}^{J} s_{ij} t_j + \sum_{k=1}^{I} f_{ki} = \sum_{k=1}^{I} f_{ik}, \qquad \forall s_i \tag{2.8}$$

$$s_{ij} = \begin{cases} 1 & s_i \in C_j \\ 0 & s_i \notin C_j \end{cases} \tag{2.9}$$

The objective of this LP is the same as the one in Section 2.1.1, which is to maximize the sum of lifetime of each set cover. Constraint 2.6 ensures the expended energy of sensor nodes is not exceed their limit. Constraint 2.7 ensures all the

targets are monitored at all times. The key difference to the LP in Section 2.1.1 is the flow conservation constraint, i.e., 2.8, which ensures the sensed data, i.e., $R\sum_{j=1}^{J} s_{ij}t_j$, can be forwarded to the sink. The fundamental problem in this LP is how to construct the collection of set covers, i.e., C_j, that maintaining complete target coverage and network connectivity.

Alfieri et al. [52] formulate the MLCC problem as a Mixed Integer Linear Programming (MILP) with the objective to maximize system lifetime. The key constraints include coverage and conservation flow. They then propose a greedy heuristic that divides time into rounds; all sensor nodes have a uniform probability to be activated in each round. If the subset of sensor nodes activated in a round ensures the required coverage and connectivity, they will be active continuously in the round. Otherwise, the heuristic discards the subset of nodes and randomly activates another set of sensor nodes. The heuristic will stop if there are no other subsets.

Zhao et al. [51] propose an algorithm that formulates the MLCC problem as an Integer Programming (IP). The objective of the IP is to maximize network lifetime subject to energy and flow conservation constraints. They then propose a greedy heuristic that divides sensor nodes into a set of disjoint set covers. The heuristic first identifies a critical target, which is the one covered by a minimal number of sensor nodes. It then chooses a sensor node to cover the critical target and determines its path to the base station for data transmission. Specifically, a sensor with a large residual energy and shorter distance to the base station has a higher priority to be selected into a path. The sensor nodes involved in this path also monitor the targets within their sensing range. The heuristic then repeatedly determines the next critical target and the sensor node to cover until all targets are covered.

Liu et al. [49] consider a scenario where each sensor node can only monitor one target at a time. Moreover, each target can only be watched by one sensor. They first use a LP to maximize system lifetime subject to energy and flow conservation constraints. The result of this LP gives the length of time that a sensor node monitors a target and the data flow between any two sensor nodes. They then

build a bipartite graph between sensor nodes and targets, which are connected by edges weighted by their corresponding monitoring time. After which, they construct set covers by extracting perfect matchings from the bipartite graph that yield the minimal monitoring time. Finally, they build a sensor node surveillance tree for each set cover based on the data flow calculated from the LP. The tree is rooted at a base station and all leaf sensor nodes are active.

Another work is by Liu et al. [50] where they extend the single coverage problem in [49] to a (h, k) coverage problem: a sensor node is able to watch h targets and each target must be watched by k sensor nodes at any time. Their approach is similar to that of [49]. They first solve an LP with the objective of maximizing the system lifetime subject to energy and conservation flow constraints. Then they build a bipartite graph that represents the time that sensor nodes monitor targets. They improve upon the algorithm in [49] to construct set covers by extracting the perfect (h, k) matchings. After that, they build a sensor node surveillance tree based on the data flow calculated from the LP to guarantee connectivity.

The reviewed MLCC protocols significantly improve network lifetime as compared to algorithms designed for the MLC problem when taking communication cost into account [49]. However, the network lifetime remains restricted by the limited battery capacity of sensor nodes. To this end, the next section reviews works using energy harvesting sensor nodes where they have the ability to replenish their battery from ambient source.

2.1.3 Duty Cycling in Energy Harvesting WSNs

This section reviews node activation protocols for sensor nodes with energy harvesting capability. The energy consumption and recharging rate of each sensor node is random. It is worth noting that sensor nodes cannot monitor targets when it is recharging. That is, it must power down to recharge. Each sensor node must ensure its residual energy is sufficient to maintain basic functionalities; e.g., compu-

tation and communications. There are two main objectives of past works on energy harvesting WNSs: maximizing the probability of individual sensor node to detect events, and maximizing the network coverage level. Here, the coverage level usually refers to a utility function that represents how well a sensing field is monitored.

2.1.3.1 Maximizing Events Detection Probability

In past energy harvesting WSNs works, researchers study WSNs whereby nodes are sparsely deployed to the extent that a target (area) may be monitored by only one sensor node. This means that each sensor node is responsible for all monitoring tasks. However, as mentioned in Chapter 1, they are unable to do so because the energy consumption rate of a sensor node is significantly higher than its recharging rate. Thus, their objective is then to maximize the probability of event detection at locations with one or more targets. This means that a sensor node needs to be activated when an event occurs.

To this end, one approach is to activate a sensor node regularly. The authors in [53] propose three node activation protocols to monitor moving objects. The first protocol, called static, requires a sensor to activate itself at a predetermined, constant time interval. In the second protocol, a sensor node activates itself based on its current energy harvesting rate and residual energy. In the last protocol, called multi-parameters heuristic, a sensor first operates the static activation protocol. Once it detects a threat, it re-calculates its activation time interval based on its current residual energy and harvesting rate.

Another approach is to divide time into fixed slots and use the history of events at target locations to activate a sensor node. The works in [54, 55] consider two scenarios: full and partial information. In the full information scenario, a sensor node is aware of the occurrence of all events even when it is asleep in the slot in which an event occurs. It then uses a Markov Decision Process (MDP) [56] to define the probability that an event will occur in subsequent time slots given past history. For the partial information case, a sensor node will be aware of an event's occurrence

only when it is active. In [54, 55], the authors formulate the partial information case as a Partially Observable MDP (POMDP) [57]. The probability of an event in the current time slot can be defined based on the set of possible past events. However, solving the POMDP problem suffers from the curse of dimensionality where the set of possible past events grows exponentially with time.

Jaggi et al. [55] consider a scenario where events are correlated in time. For example, when a fire occurs, the detected temperature at a target location may remain high for a period of time. They also consider both full and partial information cases. In the full information case, a sensor node will wake-up in the next time slot if an event occurs in the current time slot. Otherwise, it makes a decision via MDP if no event occurs. In the partial information case, they use an aggressive wakeup policy, in which, a sensor node will be activated whenever it has sufficient energy.

Ren et al. [54] extend the work in [55] and consider a general scenario where events occur randomly. In the full information case, a sensor node decides whether to activate in the current time slot based on the last event's occurrence as calculated using MDP. In the partial information case, a sensor node activates itself in a sequence of time slots where an event has the highest probability of occurring. If no event is detected at these time slots, it will use the aggressive wakeup policy of [55] until a new event is captured.

The algorithms proposed in [54, 55] both use a MDP to address the node activation problem in the full information case. In the case of partial information, the algorithm in [55] provides a general solution to maximize events detection. Moreover, the algorithms introduced in [53] is also able to activate nodes under the partial information scenario. However, these algorithms do not consider the cooperation between sensor nodes.

2.1.3.2 Maximizing Network Coverage Level

Another research direction in energy harvesting WSNs is to maximize the network coverage level. These works usually deploy sensor nodes densely. Thus, each target

may be covered by multiple sensor nodes. Past works such as [26, 58, 59] propose to use a utility function to represent network coverage level. This corresponds to the percentage of events that can be captured at target locations. Thus, the objective of these works is to maximize the overall utility and the operation time of the network.

Banerjee et al. [58] propose a centralized algorithm whereby each sensor node enters a passive state if it is fully discharged or enters the ready state when it is fully recharged. The authors assume that a central controller knows all sensor nodes' state. A utility function is used to gauge the number of ready sensor nodes. Specifically, a higher number of sensor nodes in the ready state corresponds to a lower utility. They calculate a threshold in which the number of ready sensor nodes yields the maximal utility. The central controller then uses a queue to track the time in which sensor nodes enter the ready state. If the number of ready sensor nodes in the queue reaches the computed threshold, the central controller will activate the sensor node at the front of the queue.

Tang et al. [59] also propose a centralized algorithm that divides time into equal length slots. They introduce a utility function to calculate coverage level with respect to the number of activated sensor nodes in a time slot and their location. They propose a LP with the objective to maximize the overall utility for all time slots subject to energy constraint and whether a sensor node is active in a time slot. However, if the size of the network is large, the time spent solving the LP will be too long to be practical. To this end, they propose a greedy heuristic. At each step, they select a sensor node to be activated in a given time slot that has the maximum incremental utility. The heuristic repeats until all sensors are selected.

Kar et al. [26] propose a distributed algorithm that considers the same sensor operation model as in [58] where a sensor node enters the ready state only after it is fully recharged. They propose to calculate the coverage level utility of a local area via a function corresponding to the number of activated sensor nodes and their location. Each sensor node in the ready state has to enter a decision phase periodically to exchange its location information with its working neighbors to calculate

the coverage utility. If the utility is below a predetermined value, it will activate itself. Otherwise it remains in the ready state until the next decision instant.

In summary, the centralized algorithms proposed in [58, 59] incur non-negligible communication overheads; a similar limitation also exists for algorithms in Section 2.1.1. Therefore, the algorithm introduced in [26] is suitable for large scale networks. On the other hand, the algorithms in [26, 58] have a key weakness in that a node can be activated only if it is fully charged. This weakness is addressed in [55] where they activate partially recharged sensors.

Prior works on energy harvesting WSNs aim to maximize a utility function, which represents how well targets are monitored by sensor nodes. These works have not considered the problem of ensuring all targets are monitored at all times as well as ensuring sensor nodes operate perpetually. They also do not consider the problem of maximizing a sensor node's recharging opportunities, whereby a sensor node can be activated as soon it has sufficient energy. Moreover, these works schedule multiple sensor nodes to monitor one target, which significantly reduces network lifetime.

2.2 Nodes Placement Algorithms

A key assumption of past duty cycle scheduling works is that they assume all sensor nodes are already deployed randomly around targets. As mentioned in Chapter 1, this does not guarantee sufficient number of sensor nodes around each target to ensure energy neutral operation. Moreover, unnecessary deployment of sensor nodes increases network cost. To this end, determining the locations to place sensor nodes before deriving a duty cycle is important.

The forthcoming sections review past works on the node placement problem. Specifically, it starts with the works in non-rechargeable, aka, conventional, WSNs. It then outlines the node placement works that consider the energy harvesting capability of sensor nodes.

2.2.1 Conventional WSNs

As highlighted in Chapter 1, sensor nodes in conventional WSNs have no ability to replenish their energy supply. Therefore, the goal of these works is to minimize the deployment cost given a network lifetime, or to maximize network lifetime with limited number of sensor nodes. More examples can be found in [37].

Patel et al. [60] propose to place sensor nodes to meet one of four objectives: minimum number of nodes, minimum deployment cost, minimum energy consumed or maximum lifetime. They formulate each of the objectives as a ILP. A common constraint is to ensure all targets are monitored at a given coverage level. In the objectives for minimizing number of nodes, deployment cost and minimum energy consumed, the constraints include coverage, connectivity and a desired network lifetime. In particular, in order to minimize deployment cost, they consider different nodes: sensing only nodes, relays and base station; each with different deployment cost. Moreover, they aim to minimize the consumed energy by minimizing the number of relay nodes. Lastly, they maximize network lifetime, given the number of sensor nodes, subject to coverage and flow conservation constraints.

Cheng et al. [61] first propose a non-linear program with an objective to maximize network lifetime subject to a fixed number of sensor nodes, coverage, energy and connectivity constraints. They then consider a scenario where sensor nodes with depleted energy can be replaced. They propose a function to calculate the cost of replacing an individual sensor node corresponding to its capability such as sensing, relaying and sink. They then revise the non-linear program with an objective to minimize the total cost of replacing sensor nodes.

Liu et al. [62] propose an approximation algorithm to minimize network cost, given a required network lifetime. Their algorithm also ensures coverage and network connectivity. Specifically, they first determine the locations to place sensing and relaying nodes for each target. They then remove redundant sensor nodes after all targets have a path connecting the base station/sink. They prove their algorithm

requires at most $max\{2l - m + 2, 3\}$ times more sensor nodes than the optimal solution, where m is the number of targets and l is the targets within the sensing range of the base station.

The authors of [63] aim to place the minimum number of sensor nodes to ensure targets are detected with a given probability. They divide the sensing field into cells. Each cell provides a certain 'coverage level'. The authors also define the 'miss probability' of a target to be its required event detection probability minus the 'coverage level' provided by sensor nodes covering it. They then provide a heuristic that places sensor nodes in cells with the highest miss probability until all sensor nodes have the required detection probability.

There are also works that consider placing heterogeneous sensor nodes. One aim is to minimize the total cost required to construct a WSN. The authors in [64] consider the various sensing capabilities of sensor nodes. These capabilities include monitoring temperature, sound, radioactivity and gas level. They also assume each target location has a specific sensing task. The objective is then to design an approximation algorithm to minimize the total cost, e.g., the number of sensor nodes, whilst maintaining a given coverage level for each sensing task. They prove that their algorithm has a cost of at most $1.58 + \epsilon$ times the optimal, where ϵ is a small value.

Wu et al. [65] aim to maximize the average detection probability in a sensing field by placing sensor nodes with different sensing capabilities or cost. They assume a sensor node with a large sensing range and higher detection quality to be more expensive. The design constraint is deployment budget. They show that such a problem is NP-complete and propose a genetic algorithm. This algorithm first calculates the coverage level provided by each type of sensor node at each location. After that, in order to achieve the required detection probability and budget, it uses operators such as crossover, mutation and translocation to determine the locations to place sensor nodes.

Although these works consider targets coverage, they do not consider energy

harvesting nodes. Consequently, once a node exhausts its energy, the resulting WSN is useless. Moreover, they do not consider deploying multiple sensor nodes in the same location. This is critical in energy harvesting WSNs due to varying recharging rates and duty cycling. To this end, the next section reviews node placement works that consider energy harvesting WSNs.

2.2.2 Energy Harvesting WSNs

In energy harvesting WSNs, existing works are focused on improving network coverage or connectivity. Their objectives are to minimize sensor nodes subject to a given coverage level, or ensuring connectivity. Another research direction is to use radio frequency identification (RFID) technology, where a RFID reader/sensor node is used to recharge/monitor tags/targets. These works, however, have not considered equipping RFID reader/sensor nodes with energy harvesting capability, or indeed the readers/sensor nodes have an unlimited energy supply.

Eu et at. [66] consider placing solar energy harvesting sensor nodes to monitor a one-dimensional sensing space. For example, sensor nodes placed along a railway track to monitor vibration. The objective is to minimize the number of relaying nodes to forward data from *one* source node to *one* base station/sink. However, they do not consider complete targets coverage and energy neutral operation. Moreover, it is unclear how their solution can be extended to arbitrary sensing fields.

In [67] and [68], the authors consider targets placed uniformly in a sensing field, each of which is monitored by a set of sensing nodes. They also consider the random recharging rate at different locations. The objective is to place the minimum number of relay nodes to achieve network connectivity whilst ensuring relay nodes harvest large amounts of ambient energy. They prove the problem is NP-hard and propose a 12.4-approximation solution. This solution places relay nodes at locations with large amount of data to be forwarded and has a high energy harvesting rate. However, they assume sensor nodes are already deployed around targets. Moreover, they do

not consider deploying a sufficient number of sensor nodes to achieve energy neutral operation.

In RFID technologies, e.g., [69] and [70], they consider targets coverage with two objectives. The first one is to place RFID reader/recharging nodes to ensure tags/targets have a reliable energy source. Another one is to plan the trajectory of one or more mobile RFID reader/recharging nodes to replenish the energy of tags/monitor targets. However, as mentioned, these works do not consider energy harvesting capability of RFID readers/sensor nodes.

2.3 Summary

In summary, this thesis differs from past works in the following manners:

1. Past works on the MLC problem ensure all targets are monitored by at least one sensor node throughout a WSN's lifetime. The key limitation is that the network lifetime is restricted by sensor nodes' finite battery capacity, and sensor nodes do not have the ability to recharge their battery. On the other hand, the duty cycling algorithms in energy harvesting WSNs mainly focus on maximizing events detection probability or network coverage level. They do not consider the significance of complete targets coverage. Moreover, past works do not consider the recharging opportunities of sensor nodes. To fulfill these gaps, this thesis considers the MLCEH problem with an objective to maximize network lifetime using energy harvesting sensor nodes. The key constraints include complete targets coverage, energy and recharging opportunities.

2. Past centralized coverage algorithms assume the base station/sink knows the exact battery level information of sensor nodes. However, this is not valid in practice because sensor nodes have a random recharging rate, and it is impractical for the sink to know the exact battery level of each sensor node. To this end, this thesis proposes to use stochastic programming [71] to cope

with the uncertainty battery levels of sensor nodes.

3. Past works have not considered designing a distributed algorithm for complete targets coverage using sensor nodes equipped with energy harvesting capability.

4. Similar to the MLC problem, past works on the MLCC problem do not consider the recharging opportunities of sensor nodes. To fulfill this gap, this thesis proposes a MLCCEH sub-problem to ensure all sensor nodes in the active state cover all targets whilst ensuring there is at least one path to the base station/sink.

5. Existing works on conventional WSNs node placement problem do not consider the energy harvesting capability of sensor nodes. Moreover, the works on energy harvesting WSNs neglect the importance of energy neutral operation. Thus, to fulfill this gap, this thesis considers the MEHNP-PC and MEHNP-ENCC problem to ensure energy neutral operation and complete targets coverage.

Chapter 3

Novel Algorithms for Complete Targets Coverage

This chapter addresses the MLC problem in the context of energy harvesting WSNs. The goal is to maximize a WSN's lifetime whilst ensuring *all* targets are monitored by at least one sensor node at all times. This *complete targets coverage* problem, however, has only been studied in conventional or non-rechargeable WSNs; see Chapter 2. This means they do not consider the energy harvesting capability of sensor nodes. On the other hand, existing works that solve the coverage problem in energy harvesting WSNs have only focused on maximizing the coverage probability of targets by using duty cycling [53] or prediction techniques [26]. Thus, they do not consider continuous monitoring of targets.

To address this research gap, this chapter considers the MLCEH problem where the aim is to schedule the active and sleep time of energy harvesting sensor nodes such that all targets are completely covered at all times for the longest time. To this end, two algorithms are proposed. The first one, called LP-MLCEH, relies on a LP solver. Its objective is to maximize network lifetime subject to complete targets coverage and energy constraints. Numerical results show that LP-MLCEH

doubles network lifetime when compared to similar algorithms developed for finite battery WSNs. However, it incurs high computational cost due to multiple calls to a LP solver. To this end, this chapter proposes the Maximum Utility Algorithm (MUA). Experiment results show that MUA achieves $\frac{3}{4}$ of the network lifetime of LP-MLCEH.

3.1 Network Model

This chapter considers a WSN modeled as a sensor-target bipartite graph (S, Z, E, W), where S is the set of sensors, Z is the set of targets, and E is the set of edges connecting a sensor $s_i \in S$ to one or more targets in Z. Note, s_i and z_j index sensors and targets, where $i = 1 \ldots |S|$ and $j = 1 \ldots |Z|$. Lastly, $w_{ij} \in W$ is an edge weight that represents the residual active time of sensor node s_i with respect to target z_j.

Let E_i (Joules) be the current energy of sensor node s_i, which is bounded by the battery capacity B_{max}. In addition, it has a recharging rate of E_i^r (Joule/s). Also, each sensor consumes E_i^c (Joule/s) when active. Let $Z(s_i)$ be a function that returns the set of targets covered by sensor s_i; i.e., sensor s_i covers $|Z(s_i)|$ targets. Conversely, $S(z_j)$ is a function that returns the set of sensors covering target z_j. Assume that time is divided into unequal time slots. Define $C_t \subseteq S$ to be the set cover at time slot t that is monitoring at least one target. Let $Z(C_t)$ return the set of targets covered by sensor nodes in the set cover C_t. With a slight abuse of notation, let $W(C_t)$ and $W^*(C_t)$ return the set of weights for sensor nodes in and outside of set cover C_t respectively. Let $\phi(C_t, z_j)$ be a coverage mapping function that returns one if target z_j is covered by C_t. Otherwise it returns zero. Also, $E(C_t)$ is an indicator function that returns one if the residual energy of all sensors in C_t is sufficient to provide cover throughout time slot t.

Here, an *epoch*, denoted as δ_t, is a time instant defined as one of the following: (i) when a target is not monitored by any sensor node, or (ii) when an in-active sensor node has a full battery. In the first case, any developed algorithm needs to activate

another set cover to monitor all targets. In the second case, these algorithms need to take into account sensor nodes with a full battery such that they are used to monitor targets whilst affording other nodes recharging opportunities. This means a time slot t is the duration between epoch δ_t and δ_{t+1}.

3.2 Problem Statement

The main objective is to determine the set covers and their corresponding active time such that all targets are monitored continuously. That is, determine the maximum coverage time t, where $t \in [0, \infty]$, that satisfies the following constraints: (i) $E(C_t) = 1$, and (ii) $\phi(C_t, Z) = 1$. Constraint (i) ensures all sensor nodes in cover C_t have sufficient energy. Moreover, as per constraint (ii), C_t provides complete coverage.

Note that the problem becomes NP-hard, see [27], if the problem is to determine the minimum number of sensor nodes that covers all targets and is equivalent to the minimum set cover problem. The MLCEH problem, on the other hand, aims to minimize the activation time of sensor nodes such that all targets are covered, whilst affording them ample time to recharge.

3.2.1 Analysis

Before presenting two novel solutions, an analysis of the problem at hand is first presented. Let's start with two definitions:

Definition 1. *Coverage lifetime (T) is the duration in which sensors start monitoring targets until they fail to monitor these targets due to the lack of energy.*

Definition 2. *Complete targets coverage is achieved when all targets are covered by at least one sensor at any time slot t. That is, $\phi(C_t, Z) > 0$.*

For a given target z_j and a target lifetime T, the aggregated energy expenditure of all $|S(z_j)|$ sensor nodes used to monitor z_j must be at least $E_{total}(T) = T \times E_i^c$. Let $E_h(S(z_j))$ be this total amount of energy spent monitoring target z_j. Then for

each target $z_j \in Z$, a necessary condition to achieve a lifetime of T is that it must have $E_h(S(z_j)) \geq E_{total}(T)$.

Note that $E_h(S(z_j))$ only includes the energy spent monitoring a target z_j. This means it ignores the energy used to power other components such as the microcontroller. If a sensor node is watching multiple targets, then the energy dedicated to z_j will be *proportional* to the sensor node's time watching target z_j. Let x_{ij} denote said proportion/fraction of time for sensor s_i and corresponding target z_j. Hence, the total energy used to monitor a target z_j is $E_h(S(z_j)) = \sum_{i \in S(z_j)} x_{ij} E_i^c$. We thus have the following proposition,

Proposition 1. *Perpetual coverage is achieved when* $\sum_{i \in S(z_j)} E_i^r \geq E_h(S(z_j))$ *for all* $j \in Z$.

In other words, if sensor nodes monitoring a target z_j are harvesting more energy than they are spending, then the said target will be monitored continuously. Note, the above definition does not include recharging efficiency, battery leakage and energy used to power components [25]. If these are included then the amount of harvested energy will have to be significantly higher than the energy used to monitor targets. Following Proposition 1, if one does not have perpetual coverage, then the complete coverage lifetime is governed by the target z_j with the smallest aggregated recharged energy and highest monitoring expenditure. Define,

$$Z_{min} = \left\{ z_j \mid \frac{\sum_{i \in S(z_j)} E_i^r}{E_h(S(z_j))} < 1 \right\} \tag{3.1}$$

Then the target that fails to be monitored first is,

$$z_j^* = \max_{z_j \in Z_{min}} \left[E_h(S(z_j)) - \sum_{i \in S(z_j)} E_i^r \right] \tag{3.2}$$

Hence, target z_j^* will be watched for a maximum time of

$$\frac{\sum_{i \in S(z_j^*)} E_i}{E_h(S(z_j^*)) - \sum_{i \in S(z_j^*)} E_i^r} \tag{3.3}$$

Note, the denominator of (3.3) corresponds to the shortfall in harvested energy. The maximum coverage lifetime thus does not exceed the total available energy of sensor nodes monitoring z_j^*.

According to the Proposition 1 and Equation (3.3), we see that the key to prolonging the coverage lifetime for a target is to minimize the energy/time that sensor nodes spend monitoring it. To this end, the next section presents a LP based solution and a fast heuristic with the objective to minimize the energy consumption of each sensor node whilst ensuring complete targets coverage.

3.3 Solutions

Unlike past solutions, the approaches to follow consider the recharging capability of sensor nodes. In particular, they ensure sensor nodes do not lose any recharging opportunities. This occurs when a sensor node has a full battery, and thereby is unable to store additional energy. These algorithms run at the start of each time slot, i.e., epoch. Let x_{ij}^t denote the time that sensor node i watches target j in time slot t. Hence, within a time slot, the network has complete targets coverage; see Section 3.1. Moreover, it can be seen that the lifetime of a WSN is simply the epoch δ_T in which an algorithm fails to provide complete coverage.

The following sections describe two algorithms to determine the maximum δ_T. The first one is a LP based MLCEH algorithm; i.e., LP-MLCEH. After that, Section 3.3.2 presents MUA, which selects nodes based on their residual energy level.

3.3.1 LP based MLCEH algorithm

The objective of LP-MLCEH is to maximize δ_T. To this end, it aims to minimize the energy consumption of each sensor node whilst maintaining complete targets coverage. It uses the following LP to obtain x_{ij}^t,

$$\underset{x_{ij}^t}{\text{MIN}} \sum_{t=1}^{T} \sum_{i \in S} \sum_{j \in Z} x_{ij}^t \tag{3.4}$$

Subject to:

$$\sum_{i \in S(z_j)} x_{ij}^t \geq 1, \quad \forall t = 1, \ldots, T, \forall z_j \in Z \tag{3.5}$$

$$\sum_{j \in Z(s_i)} x_{ij}^t \leq 1, \quad \forall t = 1, \ldots, T, \forall s_i \in S \tag{3.6}$$

$$B_{max} - \sum_{t=1}^{T} \left[\sum_{j \in Z(S_i)} E_i^r - E_i^c x_{ij}^t \right] \geq 0, \quad \forall s_i \in S \tag{3.7}$$

Constraint (3.5) ensures all targets are watched while constraint (3.6) ensures that each sensor node does not exceed its energy in one time slot. Lastly, constraint (3.7) ensures energy neutral operation; the total energy spent is less than the sum of a node's battery at $t = 0$, which is assumed to be full, and harvested energy. A suitable T is then determined using binary search. LP-MLCEH has $T \times (|S| + |Z|) + |S|$ constraints and $|S||Z|$ decision variables, and is computationally expensive to solve repeatedly for each T. Therefore, the next section presents a heuristic called MUA.

3.3.2 Maximum Utility Algorithm

Recall that sensor nodes are not able to store or use harvested energy when their battery is at capacity. To this end, MUA is a fast algorithm, as compared to LP-MLCEH, that minimizes energy wastage due to lost recharging opportunities. Here, utility is defined as the ratio of the number of nodes that are recharging over the number of nodes with a full battery.

Algorithm 1 shows the pseudo-code of MUA. It runs at each epoch δ_t to calculate the set of sensor nodes to be activated in the time slot t and its duration. First, every target selects a sensor node with maximal residual energy to be activated; see *Line* 5. Then δ_{t+1} is defined as the time instant when an active sensor node exhausts its energy or a non-active sensor node has a full battery, see *Line* 12-14, where L_t and R_t are the minimal operation time of activated nodes and recharging time of non-active nodes respectively. The algorithm stops if for any targets it fails to select a sensor node with sufficient energy to cover a given time slot. This means all sensor nodes surrounding a target has exhausted their energy; see *Line* 6-7.

Algorithm 1: Maximum Utility Algorithm

```
1  Input: G(S, Z, W, E)
2  Output: Cover set C_t and its activation duration t
3  C_t = ∅
4  for each target z_j ∈ Z do
5  |   Select a sensor s_i ∈ S(z_j) with the maximum w_ij ≠ 0
6  |   if No sensor node found then
7  |   |   Exit
8  |   else
9  |   |   Add s_i into C_t
10 |   end
11 end
12 L_t = MIN(W(C_t))
13 R_t = MIN{(B_max − w_ij × E_i^c) / E_i^r | w_ij ∈ W*(C_t)}
14 Return: C_t, t = min(L_t, R_t)
```

The following proposition states the run time complexity of Algorithm 1.

Proposition 2. *The run time complexity of MUA is* $\mathcal{O}(|S||Z| + |S|)$.

Proof. At each epoch, MUA will select the sensor node with the maximum weight to cover a target; see *Line* 5. In the worst scenario, this incurs $|S|$ steps for each target as it may be covered by all sensor nodes. Therefore, selecting sensor nodes to cover all the targets requires $|S||Z|$ steps; see *Line* 4-11. Moreover, determining the minimal active duration for sensor nodes in the set cover and recharging duration for sensor nodes not in the set cover, i.e., *Line* 12 and 13, require a run time complexity

of $\mathcal{O}(|C_t|)$ and $\mathcal{O}(|S| - |C_t|)$, respectively. Thus, the total run time of *Line* 12 and 13 is $\mathcal{O}(|S|)$. Consequently, the run time complexity of MUA is $\mathcal{O}(|S||Z| + |S|)$. □

As mentioned, LP-MLCEH needs to use binary search to determine T and it repeatedly solves an LP. On an Intel Core i7 CPU @ 3.5GHz with 8 G RAM computer, experiments involving 100 randomly generated networks comprising of 20 targets and 20 sensor nodes, the average running time of LP-MLCEH is 11.458 seconds while MUA only needs 0.0627 seconds.

3.4 Evaluation

The proposed algorithms are verified via simulation using the parameters of WaspMote [72], which consumes 60 mW when active and 0.2 mW when in sleep mode. All sensor nodes are equipped with an Enocean ECS310 solar cell [73]. It has a conversion rate of 10% and a recharging efficiency of 50%, which is conservative as compared to other technologies [74]. In addition, the simulation uses real solar irradiance data retrieved from Southwest Solar Research Park, Phoenix, Arizona, USA [75] on the 16-th of April 2013. Hence, for each sensor node, its recharging rate is a sinusoidal function that peaks at 12 o'clock in every 24 hours period. Other parameter values are as follows: (i) battery size, 1100 mA, (ii) consumption rate, 3.6 Joules/hour, (iii) voltage, 4V, (iv) solar panel conversion rate, 10%, and (v) recharging efficiency, 50%.

The experiments compare LP-MLCEH and MUA with the Maximum Lifetime Coverage using Disjoint Set Cover (MLC-DSC) algorithm [33], which addresses the MLC problem by dividing sensor nodes into disjoint set covers. In order to run MLC-DSC in the energy harvesting case, the simulator first runs the MLC-DSC algorithm to calculate a network lifetime. It then updates the battery of sensor nodes as per their recharging rate and reruns the algorithm again at the end of the computed lifetime. This continues until a sensor node fails to monitor a target. The experiments also compare LP-MLCEH and MUA with different rules used to select

active sensor nodes. These rules include Random, Maximum Energy First (MEF) and Maximum Target First (MTF). For the Random rule, each target is paired with a sensor node randomly. For the MEF and MTF rules, they select a sensor node with the most residual energy or covers the most number of targets, respectively, to monitor each target.

The experiment results are an average of 100 runs, each with a different topology. Each sensor node is assumed to have a maximum 76.6 hours worth of energy. However, a network may operate perpetually if it has sufficient number of sensor nodes; see Section 3.2.1. Let each time slot be one hour in length. Also, an upper bound of 30000 hours indicates that the network is operating perpetually. Recall that each sensor node is assumed to have a direct link to the sink and communication cost is negligible. Note, communication cost can be considered by scaling the available energy at each node or by reducing its recharging rate accordingly. The set of experiments investigated the following scenarios: varying number of sensor nodes, number of targets and sensing ranges.

3.4.1 Results

3.4.1.1 Node Density

The first experiment fixes the number of targets to 20 and varies the number of sensor nodes from five to 40 – both sensor nodes and targets are dispersed within a 1000 $\times$ 1000 m^2 field. Each sensor node has a uniform sensing range of 500 meters. From Figure 3.1, it can be seen that network lifetime increases rapidly when there are more sensor nodes except when using the Random rule. The reason is that as the number of sensor nodes increases, each sensor node has more opportunities to enter the sleep state, which helps increase harvested energy. For the Random rule, a target may be monitored by multiple, and hence redundant, sensor nodes. Figure 3.1 also shows LP-MLCEH achieves perpetual coverage with the minimal number of sensor nodes. On the other hand, to obtain similar performance, MUA requires approximately 10

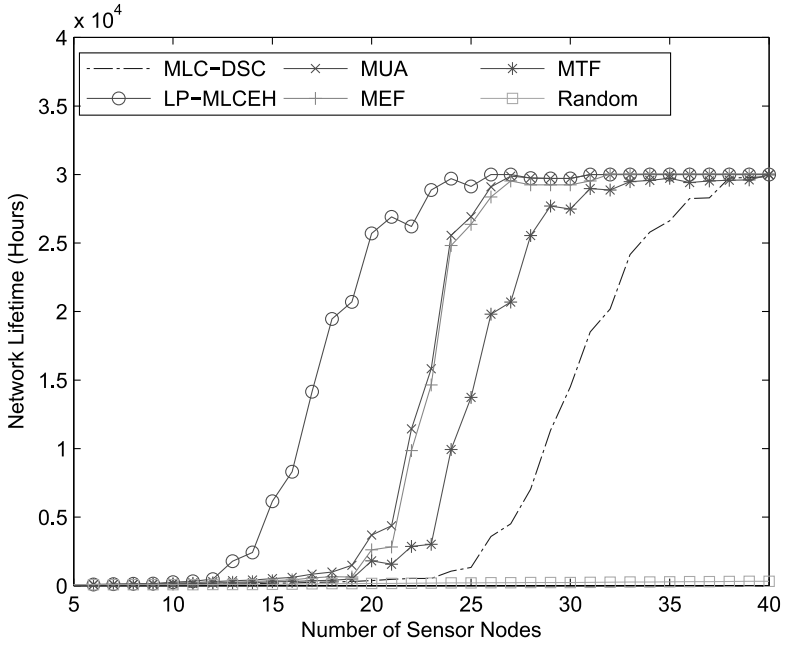


Figure 3.1: Number of Sensor Nodes vs. Network Lifetime

more sensor nodes, which is 4/3 of the number of nodes required by LP-MLCEH. It also shows that MUA needs one less sensor node than MEF because no energy is wasted due to missed energy harvesting opportunities. Furthermore, MLC-DSC requires 15 more, which is 38, sensor nodes than LP-MLCEH to achieve perpetual coverage. Therefore, MLC-DSC is not suitable for energy harvesting WSNs.

3.4.1.2 Target Density

In this experiment, the number of sensor nodes is fixed to 20 and the number of targets varies from ten to 50. Figure 3.3 shows a 20% reduction in lifetime for LP-MLCEH when the number of targets increases from 10 to 20. However, other algorithms decreased by 80% and MUA achieves the best network lifetime. In particular, the network lifetime of LP-MLCEH reduces by 30% when the number of targets increased to 40 while other algorithms recorded more than 90% reduction.

3.4.1.3 Sensing Range

The last experiment studies the effect of sensing range. All 30 sensor nodes and targets are dispersed within a 1000×1000 m^2 field. Sensor nodes' sensing range is

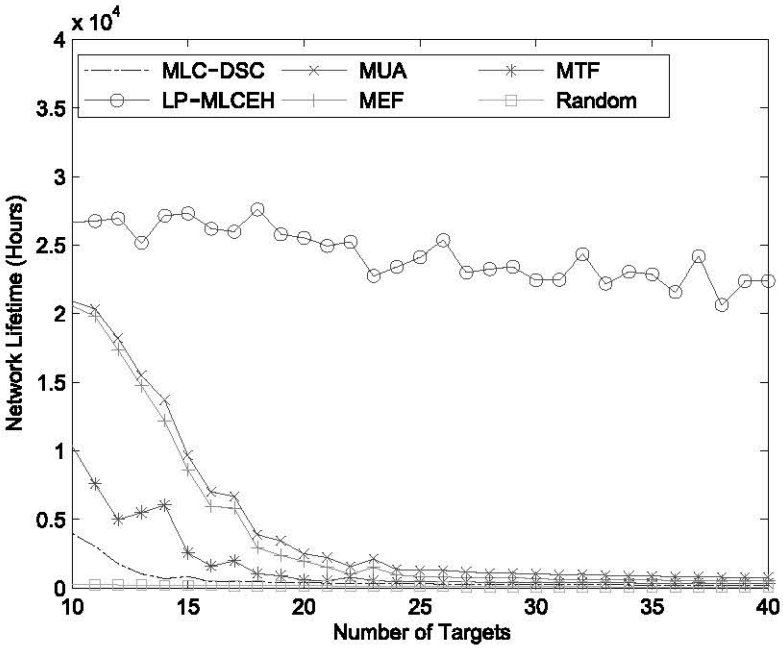


Figure 3.2: Number of Targets vs. Network Lifetime

varied from 100 to 900 meters.

From Figure 3.2, it can be seen that the network lifetime increases with different sensing range thresholds for each algorithm except for the Random rule. Upon reaching the threshold, the network lifetime calculated by these algorithms rapidly achieves perpetual coverage. For example, the network lifetime calculated by LP-MLCEH is 1000 hours when the sensing range is 250 meters and it achieves perpetual coverage when the sensing range is 400 meters. MLC-DSC is similar. It has a network lifetime of 900 hours when the sensing range is 400 meters and perpetual coverage when the sensing range is 700 meters. The reason is that once the sensing range is increasing 250 meters, there are approximately four more sensor nodes are used to monitor each target. This reduces energy consumption due to targets monitoring, and thus, sensor nodes are able to obtain perpetual coverage; see proposition 1. Figure 3.2 also shows that the network lifetime of LP-MLCEH peaked at 450 meters while MUA and MEF require the sensing range to be 500 meters. Moreover, LP-MLCEH doubles the network lifetime of MUA when the sensing range increases from 250 to 400 meters. This is because sensor nodes that run MUA and MEF become active voluntarily and may result in redundant coverage.

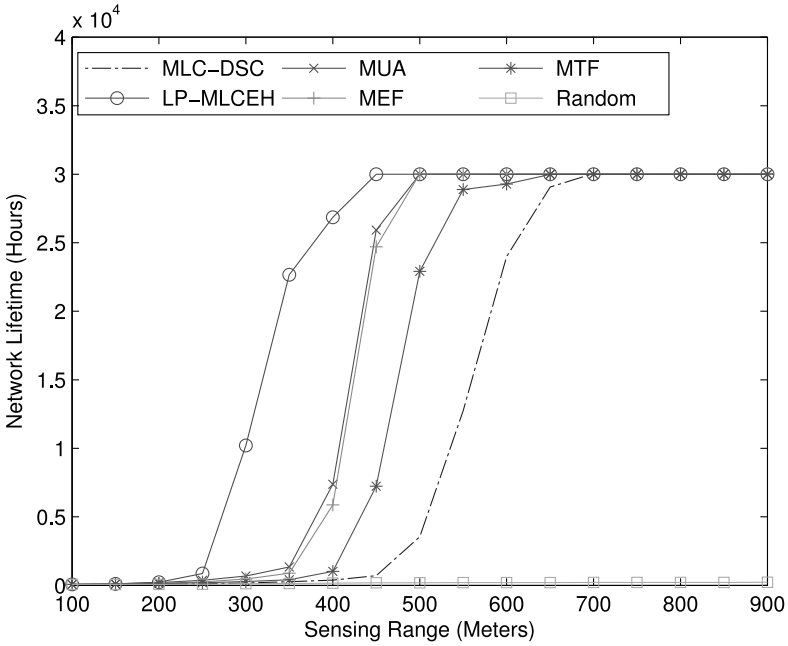


Figure 3.3: Sensing Ranges vs. Network Lifetime

3.5 Conclusion

This chapter has proposed a novel problem called MLCEH. Specifically, it has outlined two solutions that address this problem: LP-MLCEH and MUA. The first is a LP based algorithm. The results show that LP-MLCEH achieves more than twice the network lifetime of algorithms designed for finite battery WSNs. However, it is computationally expensive. This chapter therefore proposes MUA, an approach that is a few orders of magnitude faster and achieves 3/4 of the network lifetime obtained by LP-MLCEH.

A key observation is that the proposed solutions assume the sink, where the coverage algorithm is run, knows the exact battery level information of all nodes. This assumption, however, is not necessarily valid. As shown in [26], sensor nodes have random recharging rates. Therefore, the next chapter outlines a solution that considers a more realistic setting whereby the sink computes a schedule for sensor nodes using staled battery level information.

Chapter 4

Complete Targets Coverage with Uncertain Battery Levels

This chapter considers the MLCEH problem in a scenario where the battery level information known at the base station/sink is 'staled' and inaccurate. This is due to the random recharging rates experienced by sensor nodes and also delays in propagating battery level information over multiple hops to the sink; see Chapter 1. Consequently, upon receiving an activation schedule, a node may have insufficient energy to implement the schedule. Conversely, a sensor node may experience a temporary but "high" recharging rate that allows it to recharge fully. In this case, the node needs to expend its energy in order to take advantage of future recharging opportunities that in turn help prolong coverage lifetime. It is worth noting that more accurate information can be obtained if nodes coordinate their updates and send them frequently to the sink. This, however, is at the expense of precious energy, especially by nodes near the sink, which could have been used for monitoring targets. Hence, a key research question is whether one can conserve energy by reducing the frequency of updates whilst accounting for the resulting increase in uncertainty.

To this end, this chapter contains a number of contributions. First, to account

for battery level uncertainty, it presents the SP-UMLC algorithm. This algorithm solves the uncertainty problem via a two-stage SP with the goal of minimizing the activation time of sensor nodes. It then solves the SP in the SAA framework due to the exponential number of scenarios [28]. Secondly, this chapter shows a modified algorithm based on the LP-MLCEH algorithm proposed in Chapter 3 to incorporate a penalty for nodes with a high battery level; the new formulation is denoted as LP-MLCEH-P. In experiments where LP-MLCEH-P uses accurate battery level information, a theoretical benchmark that requires the sink to take a snapshot of the current battery level at each node at a time point, SP-UMLC achieves 80% of the coverage lifetime attained by LP-MLCEH-P.

4.1 Network Model

In this chapter, a WSN is modeled as a sensor-target bipartite graph (S, Z, E, W). Here, S is the set of sensors, Z is the set of targets, and E is the set of edges connecting a sensor $s_i \in S$ to one or more targets in Z. Note, the notations s_i and z_j are used to index sensors and targets, where $i = 1 \ldots |S|$ and $j = 1 \ldots |Z|$. Let $Z(s_i)$ and $S(z_j)$ be a function that returns the set of targets covered by sensor s_i and the set of sensors covering target z_j respectively. Assume that time is divided into intervals, indexed by t. Each interval refers to a *time slot*. Define $C_t \subseteq S$ to be the set of nodes providing complete coverage at time slot t. With a slight abuse of notation, let $Z(C_t)$ be a function that returns the set of targets covered by sensors C_t. Let $\phi(C_t, z_j)$ be a coverage mapping function that returns one if target z_j is covered by C_t, otherwise it returns zero. Also, $E(C_t)$ is an indicator function that returns one if the residual energy of all sensors in C_t is sufficient to cover time slot t. Let E_i (Joules) denote the level of sensor node s_i's rechargeable battery, which is bounded by B_{max}. To safeguard against an imprecise schedule, explained later, each sensor node i has a fixed non-rechargeable battery reserve, denoted as R_i. This reserve is only drawn upon if there is a shortfall in energy.

In the following sections, E_i^t refers to the *current* battery level of sensor node s_i, and a subsequent update as E_i^{t+1}. The uncertainty in battery level is modeled as follows. Let u represent the variation in recharging rates, and $\gamma(u)$ be a random value generated from a standard normal distribution in the range of $1-u$ to $1+u$. At E_i^{t+1}, the battery level of node i is

$$E_i^{t+1} = E_i^t - E_i^c x_i^t + E_i^r(1+\gamma(u)) \tag{4.1}$$

where E_i^r is the recharging rate of sensor node s_i, which is governed by a known probability distribution. The term E_i^c and x_i^t refer to s_i's consumption rate when active and its activation time at time slot t. Sensor nodes are assumed to be able to sense omni-directionally and thus monitor one or more targets with equal energy consumption rate. In subsequent sections, in terms of battery level information, E^{t+1} refers to the *accurate*, which is the battery level at sensor nodes. The information at the sink, however, is *staled*, denoted as E^t.

4.2 Problem Statement

This section presents the *deterministic* version of the complete target coverage problem. The goal is to determine the maximum coverage time T, where $T \in [0, \infty]$, that satisfies the following constraints: (i) $E(C_t) = 1$, and (ii) $\phi(C_t, Z) = 1$. This problem becomes NP-hard, see [27], if the aim is to determine the minimum number of sensor nodes that covers all targets. However, the problem in this chapter is to seek the minimum activation time for sensor nodes such that all targets are covered, whilst affording them ample time to recharge. Mathematically, the problem can be modeled as the following LP, where the objective is to minimize each sensor node i's active time to monitor a target j; i.e., x_{ij}.

$$\text{MIN} \sum_{i \in S} \sum_{j \in Z} x_{ij} \tag{4.2}$$

Subject to:

$$\sum_{i \in S(z_j)} x_{ij} \geq 1, \quad \forall z_j \in Z \tag{4.3}$$

$$\sum_{j \in Z(s_i)} x_{ij} E_i^c \leq E_i, \quad \forall i \in S, \tag{4.4}$$

Constraint (4.3) ensures each target is watched for at least one time slot. Constraint (4.4) ensures the total energy expenditure is within limit. Recall that each sensor node i is able to sense omni-directionally. However, term $\sum_{j \in Z(s_i)} x_{ij}$ does not take this fact into account. To this end, in the evaluation, both $\sum_{j \in Z(s_i)} x_{ij}$ and $\sum_{j \in Z} x_{ij}$ are divided by $|Z(s_i)|$ to yield the correct activation time.

Notice that a key assumption of constraint (4.4) is that the scheduler/sink is aware of the current energy level of each node. However in practice, due to random recharging rates and delay related to multi-hop communications, when sensor nodes receive their respective x_{ij} value, they may find that the computed x_{ij} value to be infeasible because the scheduler/sink used staled information to compute the schedule.

4.3 The Approach

This section outlines the SP based approach. It first provides a brief introduction to two-stage SP [71]. In the first stage, a decision is made based on the "current" battery level of nodes. In the second stage, actual battery levels become available, which require recourse actions to be carried out if the decision made in the first stage is inadequate; e.g., the scheduled active time exceeds a node's energy constraint, and thus it has to draw energy from its reserve as a recourse. Mathematically, the first stage is as follows,

$$\min_{x \in X} \{g(x) := c^T x + \mathbb{E}[Q(x, \xi)]\} \tag{4.5}$$

Then, given the first stage decision x and random vector $\xi = (q, T, W, h)$, the second stage problem is as follows,

$$Q(x, \xi) = \min_{y}\{q^T y \mid Tx + Wy \leq h\} \tag{4.6}$$

Here, the decision variable y is the recourse action to be undertaken in order to meet the budgetary constraint h. Note, the actual value and interpretation of the components in ξ, which can be fixed or random, are application specific.

In the first stage, the scheduler first determines the set of sensor nodes and their active time based on E_i^t. The second stage uses E_i^{t+1}, which is governed by random recharging rates. Hence, the goal of the second stage is to minimize the expected recourse cost. In order to ensure the scheduler/sink preferentially activates sensor nodes with a full battery, a penalty coefficient ω_i is added to each variable x_{ij} in the objective function. This coefficient conversely reflects the i-th node's residual energy level. For example, if sensor node i's battery is at $100\%, 90\%, \ldots, 0\%$ capacity, then ω_i will be set to $1, 2, \ldots, 10$ respectively.

The earlier LP formulation for the problem, see Section 4.2, can be rewritten to consider random battery levels and recharging opportunities. In the first stage, it has,

$$\text{MIN} \sum_{i \in S} \sum_{j \in Z} \omega_i x_{ij} + \mathbb{E}_\rho[Q(x_{ij})] \tag{4.7}$$

Subject to:

$$\sum_{i \in S(z_j)} x_{ij} \geq 1, \quad \forall z_j \in Z \tag{4.8}$$

$$\sum_{j \in Z(s_i)} x_{ij} E_i^c \leq E_i^{t-1}, \quad \forall i \in S, \tag{4.9}$$

$$\sum_{j \in Z(s_i)} x_{ij} \geq 0, \quad \forall i \in S, \tag{4.10}$$

The main changes are to (i) the objective function, which now considers the un-

certainties caused by the varying battery levels, as described by the probability distribution ρ, and (ii) constraint (4.9), which reflects the sink's record of nodes' *current* battery level.

The second stage problem, i.e., $Q(x_{ij})$, is similar. Let y_{ij} be the activation time taken as a recourse in solving the second stage problem, and also corresponds to a sensor node drawing from its battery reserve. Hence, in order to discourage its use, a high penalty ω' is added to each y_{ij}, where $\omega' \gg 10$. Specifically,

$$Q(x_{ij}) = \text{MIN} \sum_{i \in S} \sum_{j \in Z} \omega' y_{ij} \tag{4.11}$$

Subject to:

$$\sum_{j \in Z(s_i)} (x_{ij} - y_{ij}) E_i^c \leq E_i^t, \quad \forall i \in S, \tag{4.12}$$

$$\sum_{j \in Z(s_i)} y_{ij} \leq R_i, \quad \forall i \in S, \tag{4.13}$$

$$\sum_{j \in Z(s_i)} y_{ij} \geq 0, \quad \forall i \in S, \tag{4.14}$$

Note that x_{ij} is determined by the first stage problem. The term E_i^t in constraint (4.12) is a realization of sensor node i's battery level at the sink. Each realization is generated from a probability distribution function. Also, the term y_{ij} models the recourse taken given x_{ij} and E_i^t. Constraint (4.13) ensures recourse actions are limited by nodes' battery reserve. In the experiments, if y_{ij} exceeds node i's battery reserve, then the simulation ends and records the resulting lifetime.

The main difficulty in solving the SP problem is the number of battery levels each node has; so called 'scenarios'. Assuming b discrete battery levels for each node, then a WSN with 50 nodes has a total number of b^{50} scenarios! To this end, the sample average approximation (SAA) method is employed, which uses Monte Carlo simulation [28] to yield a sample average estimate of the expected recourse

cost. In particular, it estimates $\mathbb{E}_\rho[Q(x_{ij})]$ as follows,

$$\frac{1}{N}\sum_{j=1}^{N} Q(x_{ij}, \xi^j) \tag{4.15}$$

where ξ^j is a generated sample represented as a vector of dimension $|S|$ with component E_i^t, and N is the total number of required samples; explained further below.

In words, SAA requires solving (4.11)-(4.14) for each sample ξ^j, with each result weighted 1/N. To ensure the second stage always has a solution, which is a precondition for applying SAA, see [28], the value range of y_{ij} is set to be unbounded.

To measure the quality of the solution generated by SAA, one can use the method developed in [28]. Specifically, given a solution $\hat{x}^*$, the optimality gap is defined as,

$$\hat{z}_{N'}(\hat{x}^*) - \bar{z}_N \tag{4.16}$$

Now define $\hat{x}^*$, $\hat{z}_{N'}(.)$ and $\bar{z}_N$. Let $\bar{z}_N$ denote a solution to the SP problem computed using SAA. The SAA proceeds by generating M candidate solutions, and denote the k-th objective value as $\bar{z}_N^k$ and the corresponding vector of solutions, i.e., x_{ij} by $\hat{x}^k$. The average of these M solutions is,

$$\bar{z}_N = \frac{1}{M}\sum_{m=1}^{M} z_N^m \tag{4.17}$$

Next, for a given solution $\hat{x}$, i.e., nodes' wake-up time, its corresponding $\hat{z}_{N'}(\hat{x})$ is calculated as follows,

$$\hat{z}_{N'}(\hat{x}) = c^T\hat{x} + \frac{1}{N'}\sum_{j=1}^{N'} Q(\hat{x}, \xi^j) \tag{4.18}$$

where c is a vector of all ones, and $N' \gg N$. Lastly, $\hat{x}^*$ is defined as,

$$\hat{x}^* = \underset{x^k,\, k\in[1,M]}{\arg\min} \{\hat{z}_{N'}(x^k)\} \tag{4.19}$$

In the evaluation, see next section, it discretizes nodes' battery to 100 levels and pick a M and N value that ensures the gap, see Equation (4.16), is within 1% of the average objective value $\bar{z}_N$.

4.4 Evaluation

This section presents the performance of the proposed two stage SP-UMLC algorithm with different uncertainty level $\pm u$; see Equation (4.1). The experiments use the parameters of WaspMote [72], which consumes 60 mW when active and 0.2 mW when in sleep mode. All sensor nodes are equipped with an Enocean ECS310 solar cell [73]. This solar cell has a conversion rate of 10% and a recharging efficiency of 50%, which is conservative as compared to other technologies [74]. In addition, the experiments use real solar irradiance data retrieved from Southwest Solar Research Park, Phoenix, Arizona, USA [75] on the 16-th of April 2013. Hence, for each sensor node, its recharging rate is a sinusoidal function that peaks at 12 o'clock in every 24 hours period. Other parameter values are as follows: (i) battery size, 1100 mA, (ii) consumption rate, 3.6 Joules/minute, (iii) voltage, 4V, (iv) solar panel conversion rate, 10%, and (v) recharging efficiency, 50%. For the SP-UMLC algorithm, it allocates 10% of the battery capacity of sensor nodes as non-rechargeable back-up at the start of each experiment.

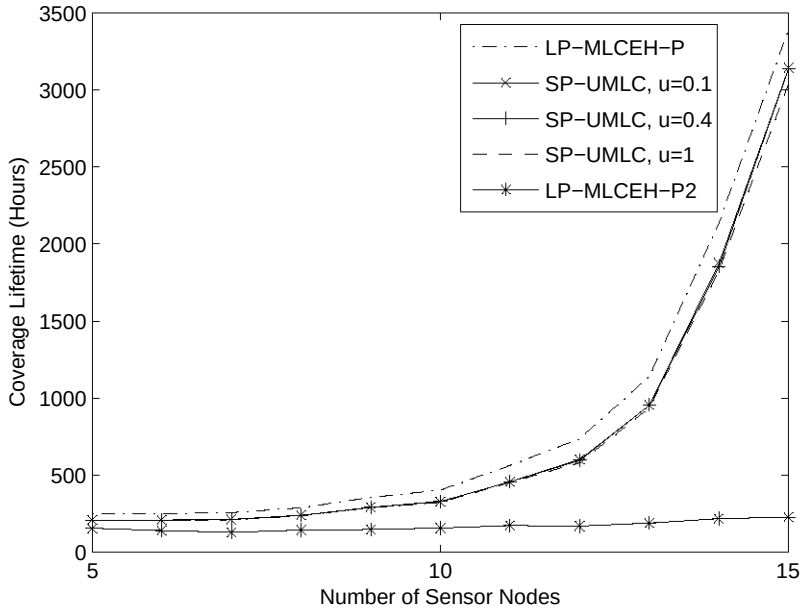
In the sequel, a comparison between SP-UMLC and LP-MLCEH [76] is presented; the latter is a theoretical approach that has accurate battery level information. It assumes an oracle exists that could gather this information without incurring any energy cost. Also, as shown in Chapter 3, the LP formulation of LP-MLCEH neglects recharging opportunities. Thus, a penalty is added to each x_{ij} in the objective function of LP-MLCEH, similar to the SP in Section 4.3, so that the LP solver preferentially activates nodes with a full battery. The revised LP is called as LP-MLCEH with penalty or LP-MLCEH-P. For comparison against SP-UMLC, the experiments contain the coverage lifetime of LP-MLCEH-P based

on staled information, which is labeled as LP-MLCEH-P2.

In the experiments, sensor nodes are dispersed within a 100×100 m^2 field. All sensor nodes have a uniform sensing range of 50 meters and a maximum 76 hours worth of energy. Moreover, sensor nodes have a different average recharging rate, which is reasonable as the recharging rate of sensor nodes is dependent on their location; e.g., sensor nodes obstructed by foliage will inevitably have a lower recharging rate [77]. In the experiments, both the number of samples and scenarios are set to five, which is found sufficient to yield an optimality gap of less than 1%.

4.4.1 Results

The first experiment is to compare the *average* coverage lifetime, i.e., a network lifetime that maintains complete targets coverage, of LP-MLCEH-P, LP-MLCEH-P2 and SP-UMLC when the uncertainty is $u = 0.1$, $u = 0.4$ and $u = 1$. In this experiment, the number of targets is fixed to 20 and number of sensor nodes varies from five to 15. The results are an average of 200 runs, each with a different randomly generated topology. Referring to Figure 4.1, the coverage lifetime of LP-MLCEH-P and SP-UMLC increases rapidly from 200 hours to more than 3000 hours. The reason is because sensor nodes have more opportunities to be in the sleep state and harvest energy. On the other hand, LP-MLCEH-P2, which activates sensor nodes using staled information, has poor coverage lifetimes. Indeed, SP-UMLC outperforms LP-MLCEH-P2 and achieves 80% of the average coverage lifetime attained by LP-MLCEH-P even though it uses staled information. Another observation is that the average coverage lifetime of SP-UMLC when $u = 0.1$ and $u = 0.4$ is very close but reduces by 350 hours when uncertainty is one. This is due to the significant variation in battery levels, which leads to unnecessarily long active times, leading to energy wastage. Next, the experiment investigates the variation in coverage lifetimes. It plots the Probability Density Function (PDF) of coverage lifetimes when the number of sensor nodes is 12; see Figure 4.2. The result is similar for other node



]

Figure 4.1: Sensor node density versus coverage lifetime

numbers. It can be seen that 90% of the recorded lifetimes are within 450 hours to 470 hours when uncertainty is 0.1. However, this percentage reduces to 20% when $u = 1$.

4.5 Conclusion

This chapter is the first to consider the uncertain battery level when solving the complete targets coverage problem. The proposed stochastic programming based solution is shown to be within 80% of the theoretical coverage lifetime, and thus is a promising solution that addresses the trade-off between uncertainties and energy consumption, where sensor nodes send updates to the sink frequently in order to ensure accurate battery level information.

A key limitation of the algorithms presented in Chapter 3 and 4 is that they require a central controller, which gathers information from sensor nodes, compute a schedule before informing sensor nodes their respective active/sleep schedule. This causes sensor nodes to expend considerable amounts of energy due to high communication cost. Moreover, synchronizing all sensor nodes is also a critical challenge. To

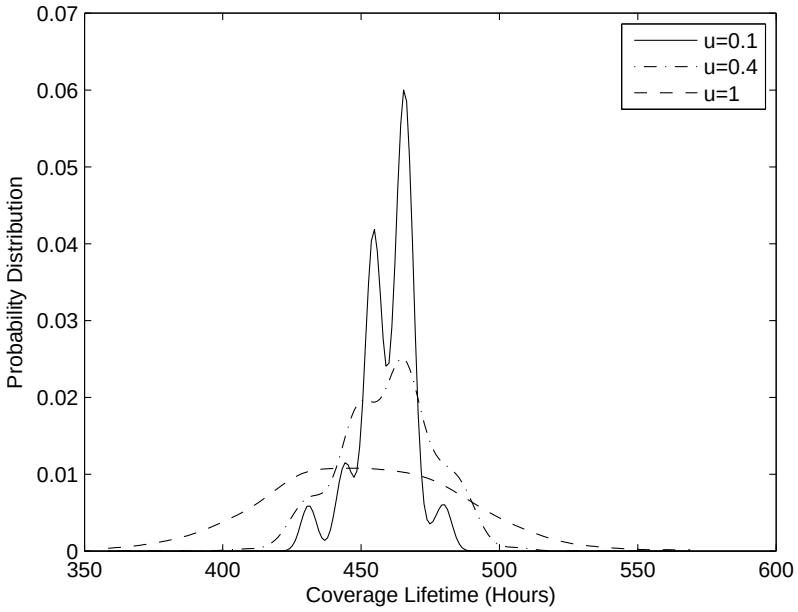


Figure 4.2: Coverage lifetime PDF of SP-UMLC under different uncertainties

this end, the next chapter outlines a distributed algorithm for the MLCEH problem.

Chapter 5

A Distributed Solution for the MLCEH Problem

This chapter considers designing a distributed solution for the MLCEH problem. The aim is to remove the reliance on a central controller. In addition, *all* nodes are synchronized locally; i.e., with their immediate neighbors only. As there is no controller, sensor nodes need to make decision based on the information from their two hop neighbors. This thus reduces signaling overheads and communication cost.

The solution presented in this chapter is called Maximum Energy Protection (MEP) algorithm. The main idea is to replace on-duty sensor nodes with those currently in the sleep state that have a higher energy level. Also, it needs to ensure that sensor nodes do not lose any recharging opportunities. This occurs when a sensor node has a full battery, and is therefore unable to store additional energy. In addition, an *eligibility test* is proposed to reduce redundant coverage, and thereby, minimize energy wastage.

5.1 Network Model

The energy harvesting WSN under consideration is comprised of sensor nodes placed on an Euclidean plane. Sensor nodes can either be in the active (on-duty) or sleep (off-duty) state. The index s_i and z_j refer to a sensor and a target, where $i = 1 \ldots |S|$ and $j = 1 \ldots |Z|$. Let E_i (Joules) be the current energy of sensor node s_i, which is bounded by B. Each sensor node s_i has a recharging rate of E_i^r (Joule/s) and an energy consumption rate of E_i^c (Joule/s). Define $Z(s_i)$ to be a function that returns the set of targets covered by sensor s_i. Conversely, $S(z_j)$ is a function that returns the set of sensor nodes covering target z_j.

All sensor nodes have the same sensing range with a communication range that is twice their sensing range. Let $N(s_i)$ denote the *neighbors* of s_i; i.e., nodes in $N(s_i)$ are those within the communication range of node s_i. Given the definition of *coverage lifetime* and *complete targets coverage* presented in Chapter 3, we have the following proposition:

Proposition 3. *The neighbors of sensor node s_i are the sensor nodes that cover the same targets as s_i.*

Proof. Assume that the sensing range of sensor node s_i is r and its communication range is $2r$. If a target z_j is within the sensing range of s_i, then let $d_{ij} \leq r$ be the distance between s_i and z_j. Assume that another sensor node s_k also covers target z_j and its distance is $d_{kj} \leq r$. Therefore, by the triangle inequality, the distance between sensor node s_i and s_k is $d_{ij} + d_{kj} \leq 2r$, which does not exceed the communication range of these two nodes. □

Define a *reshuffle phase* to be a time period of length ϕ where all involved sensor nodes decide whether to go to sleep. The length of ϕ is a constant and much less than a sensor node's total battery lifetime; i.e., $\phi \ll \frac{B}{E_i^c}$. An *epoch*, denoted as δ_i, is the time instant when sensor node s_i starts its reshuffle phase. Thus, the reshuffle phase of sensor node s_i spans time δ_i to $\delta_i + \phi$.

Here, decisions are made following an *off-duty* rule, which is commonly used to turn off overlapping sensor nodes [40, 46, 48]. The rule states: *if all targets within the sensing range of s_i are covered by a subset of its neighbors $N(s_i)$, sensor node s_i can turn itself off without reducing the overall targets coverage. Otherwise, s_i goes into the active state.*

At the beginning of each reshuffle phase, all sensor nodes involved first exchange their next epoch, i.e., δ_i, and targets information. Then, each sensor node performs the off-duty rule after a delay T_i, which is defined as follows,

$$T_i = (1 - \frac{E_i}{B}) \times \phi, \quad \forall i \in S \tag{5.1}$$

where E_i is the current battery level of sensor node s_i. That is, Equation (5.1) ensures that sensor nodes with a higher residual energy will make decision earlier.

A coverage hole may exist if more than one sensor node makes decision simultaneously [36]. Consider two sensor nodes A and B that monitor the same targets. If A and B have the same current energy level during the reshuffle phase, they will both go to sleep by assuming another node is active. To this end, Equation (5.1) can be revised to $T_i = (1 - \frac{E_i}{B}) \times \phi + \tau$, where τ is a random value less than ϕ. At the end of the reshuffle phase, only a subset of sensor nodes is in the active state to carry out the sensing task while other sensor nodes enter the sleep state to save energy and recharge.

5.2 Solution

This section presents the solution for the DMLCEH problem. Recall that an energy harvesting sensor node needs to consider recharging opportunities. In particular, it needs to be activated when its battery reaches capacity. At such time, it volunteers to monitor target(s). However, if it voluntarily wakes up whilst all targets within its sensing range are already monitored by other sensor nodes, then it will have wasted

its energy. To this end, an *eligibility test* is proposed to allow an on-duty sensor node to choose a subset of sleeping neighbors that are able to cover all its targets. When these sleeping neighbors are in the active state, the on-duty sensor node can turn itself off without reducing network coverage.

Henceforth, the next section defines the *eligibility test* used by sensor nodes. Then Section 5.2.2 presents the details of MEP; an algorithm that allows sensor nodes to form a minimal set cover to monitor all targets.

5.2.1 Eligibility Test

The eligibility test aims to determine whether a sensor node in the sleep state is *valuable* and hence it is to be activated. Here, valuable means a sleeping sensor node is able to form a subset with other sensor nodes that cover all targets monitored by an on-duty sensor node. Each on-duty sensor node needs to perform an eligibility test for all its sleeping neighbors.

Before delving into the details, we need a few key definitions. Let sensor node s_i be in the active state with an epoch of δ_i. Note, an active sensor node's epoch corresponds to the time when its battery is depleted. On the other hand, the epoch of a sensor node that is asleep is set to be the time when its battery is fully recharged. Let $Z(N(s_i))$ denote the targets covered by s_i's neighbors, and $\Delta_{N(s_i)}$ contains the epoch of these neighbors; i.e., $\Delta_{N(s_i)} = \{\delta_j \mid \forall j \in N(s_i)\}$. The set $subN(s_i)$ contain neighbors of s_i that together cover all targets monitored by s_i; i.e., $Z(s_i) \subseteq Z(subN(s_i))$ where $subN(s_i) \subseteq N(s_i)$.

Let s_j be a neighbor of s_i. When s_j reaches its epoch and enters the reshuffle phase, both s_j and s_i will exchange epoch and targets information. Sensor node s_i then decides whether it requires s_j to update its epoch δ_j by carrying out one or more of the following parts of the eligibility test:

1. If $s_j \in subN(s_i)$ and $\delta_j > \delta_i$, then s_j passes this part and the eligibility test ends. Otherwise, it continues with the next part.

2. If s_j is in $subN(s_i)$, then s_j passes. Otherwise, s_j fails the eligibility test.

If s_j passes the first part, then s_i will exhaust its energy before s_j reaches its epoch. In this case, s_i will require s_j to update its epoch to δ_i. This is to ensure s_j enters the reshuffle phase when s_i exhausts its energy.

If s_j fails the first part, meaning its battery will be at capacity before s_i enters the reshuffle phase, and thus it needs to activate itself as soon as possible to avoid losing recharging opportunities. A key consideration, however, is redundant coverage. Thus, both s_i and s_j need to enter the reshuffle phase at the same time to decide whether s_i can turn itself off using the off-duty rule. At such time, all other sensor nodes in $subN(s_i)$ also need to enter the reshuffle phase. This is because s_j may only cover some of the targets monitored by s_i. Therefore, both s_i and s_j set their epoch to MIN ($\Delta_{subN(s_i)}$). Consequently, s_i will require subsequent sensor nodes in $subN(s_i)$ that perform the eligibility test to update their epoch to MIN ($\Delta_{subN(s_i)}$). This is to ensure all sensor nodes in $subN(s_i)$ as well s_i enter the reshuffle phase together.

When s_j has two or more on-duty neighbors, it may be required to update its epoch multiple times. In this case, s_j will set its epoch to the minimum one. This is to ensure s_j enters the reshuffle phase before any on-duty neighbor exhausts its energy.

If s_j fails both parts of the eligibility test, then sensor nodes s_i and s_j retain their own epoch. This is because s_j is not in the set $subN(s_i)$ and hence, it is not able to cover any targets monitored by s_i. The next subsection will introduce the procedure that constructs $subN(s_i)$ and a novel distributed algorithm to solve the DMLCEH problem.

5.2.2 Maximum Energy Protection (MEP) Algorithm

Algorithm 2 shows the pseudo-code of MEP. A sensor node i runs MEP upon reaching an epoch. MEP then returns the next epoch δ_i. Initially, all sensor nodes enter

the reshuffle phase together and decide their status based on the off-duty rule; see *Line* 6 and 17. This ensures redundant sensor nodes enter sleep state. Note, a sensor node will broadcasts its epoch immediately once it wakes up; see *Line* 4. If sensor node s_i decides to go into the sleep state, it sets its epoch to $\delta_i = \frac{B-E_i}{E_i^r}$ and broadcasts a 'TURN OFF' message containing its epoch and targets; see *Line* 6-8. It then listens until time $\delta_i + \phi$ to receive any 'SET EPOCH' message, which will be explained later, before turning off all functions; see *Line* 9-12. On the other hand, sensor nodes that decide to be active will set their epoch to their current battery lifetime and initialize the set $subN(s_i)$; see *Line* 17-24. This set contains only the sensor nodes that are able to cover at least one target monitored by s_i.

An active sensor node, say s_i, after receiving a 'TURN OFF' message from a neighbor, say s_j, carries out the eligibility test. If s_j passes the first part of the eligibility test, s_j updates its epoch to that of s_i. Sensor node s_i then replies with a 'SET EPOCH' message containing δ_i; see *Line* 26-27. Otherwise, if s_j has a smaller epoch than s_i and is in the set $subN(s_i)$, meaning s_j passes the second part of the eligibility test, sensor node s_i then updates its epoch to $MIN(\Delta_{subN(s_i)})$. After that, s_i replies with a 'SET EPOCH' message containing the new epoch; see *Line* 29-31. If s_j fails both parts of the said eligibility test, s_i does not reply, meaning s_j retains its original epoch.

Recall that a sensor node receiving two or more 'SET EPOCH' messages will set its epoch to equal the minimum one; see Section 5.2.1. However, it may cause redundant coverage when this node reaches its epoch and decides to turn on. This is because there are other on-duty nodes with a larger epoch. For example, if s_i receives two 'SET EPOCH' messages from its two on-duty neighbors A and B with epoch $\delta_A = 5$ and $\delta_B = 8$, sensor node s_i will set its epoch to five and enter the reshuffle phase at such time. When s_i reaches its epoch and enters the active state, the targets covered by both s_i and B are monitored by two sensor nodes for three units time, which is a waste of energy. This is because sensor node B has three units time remaining before it enters its next reshuffle phase. Therefore, in order

to reduce redundancy, a sensor node that activates itself will broadcast a 'CHECK REDUNDANT' message to its on-duty neighbors. When an on-duty sensor node receives said message, it enters the reshuffle phase immediately to decide its status; see *Line* 34-38.

5.3 Evaluation

The evaluation uses the parameters of the WaspMote [72] platform, which consumes 60 mW when in the active state and 0.2 mW when sleeping. Other parameter values can be found in Table 5.1. All sensor nodes are equipped with an Enocean ECS310 solar cell [73] to harvest solar energy. This solar cell is assumed to have a conversion rate of 10% and a recharging efficiency of 50%, which is conservative as compared to other technologies [74]. In addition, the experiments use real solar irradiance data retrieved from Southwest Solar Research Park, Phoenix, Arizona, USA [75] on the 16-th of April 2013. All experiments are simulated using Matlab running on an Intel Core i7 CPU @ 3.5GHz with 8 G RAM computer.

The experiments compare MEP to Coverage-Preserving Node Schedule (CPNS) [40] and Deterministic Energy-Efficient Protocol for Sensing (DEEPS) [41]. CPNS operates in equal length rounds, in which all sensor nodes decide their status at the beginning of each round. On the other hand, DEEPS allows each sensor node to decide its status according neighbor knowledge. However, a sensor node using DEEPS will operate until it exhausts all its energy. All sensor nodes are assumed to be stationary and randomly located on a square area. Note that, the experiments do not consider the energy consumed due to sensing and forwarding of data. However, this consumption can be considered by scaling the available energy at each node for monitoring targets or by reducing the recharging rate of sensor nodes. In the evaluation, each sensor is assumed to have a timer to record its epoch. The said timer is able to trigger a sensor node that is in the sleep state to enter the reshuffle phase.

Algorithm 2: Pseudocode of MEP

```
1  Input: B, E_i, E_i^r, E_i^c
2  Output: δ_i
3  WakeUp
4  Send δ_i to nodes in N(s_i)
5  Wait for T_i = (1 − E_i/B) × φ + τ
6  if all targets in Z(s_i) are monitored by other sensor nodes then
7  |  Set δ_i = (B − E_i)/E_i^r
8  |  Send 'TURN OFF' message to nodes in N(s_i)
9  |  while current time is less than δ_i + φ do
10 |  |  if Received a 'SET EPOCH' message then
11 |  |  |  Update δ_i
12 |  |  end
13 |  end
14 |  Change state to off-duty
15 |  SetWakeUp(δ_i)
16 else
17 |  if there are un-monitored target(s) in Z(s_i) then
18 |  |  Change state to on-duty and set δ_i = E_i/E_i^c
19 |  |  subN(s_i) = ∅
20 |  |  for all s_h ∈ N(s_i) do
21 |  |  |  if Z(s_i) ∪ Z(s_h) ≠ ∅ then
22 |  |  |  |  subN(s_i) = subN(s_i) + s_h.
23 |  |  |  end
24 |  |  end
25 |  |  Upon receiving a 'TURN OFF' message from s_j
26 |  |  if δ_i < δ_j AND s_j ∉ subN(s_i) then
27 |  |  |  Send a 'SET EPOCH' message to s_j
28 |  |  else
29 |  |  |  if s_j ∈ subN(s_i) then
30 |  |  |  |  δ_i = MIN (Δ_subN(s_i))
31 |  |  |  |  Send 'SET EPOCH' message to s_j
32 |  |  |  end
33 |  |  end
34 |  |  Upon receiving a 'CHECK REDUNDANT' message
35 |  |  if All targets Z(s_i) are monitored by other sensor nodes then
36 |  |  |  Update E_i
37 |  |  |  Goto line 7 .
38 |  |  end
39 |  |  GoSleep(δ_i)
40 |  end
41 end
```

Parameters	Value
Battery size	1100 mA
Consumption rate	60 mW
Average recharge rate	16 mW
Voltage	4 V
Solar panel conversion rate	10 %
Recharging efficiency	50 %

Table 5.1: Simulation Parameters

The experiment results are an average of 50 runs, each with a different topology. Each sensor node has a maximum 76.6 hours worth of energy. However, a network may operate perpetually if there are sufficient number of sensor nodes [25]. The network lifetime is upper bounded to 3000 hours, at which time the network is said to operate perpetually. The experiments conducted investigate the impact of the following parameters: *target density*, *node density* and *sensing range*. In each experiment, the following metrics are collected:

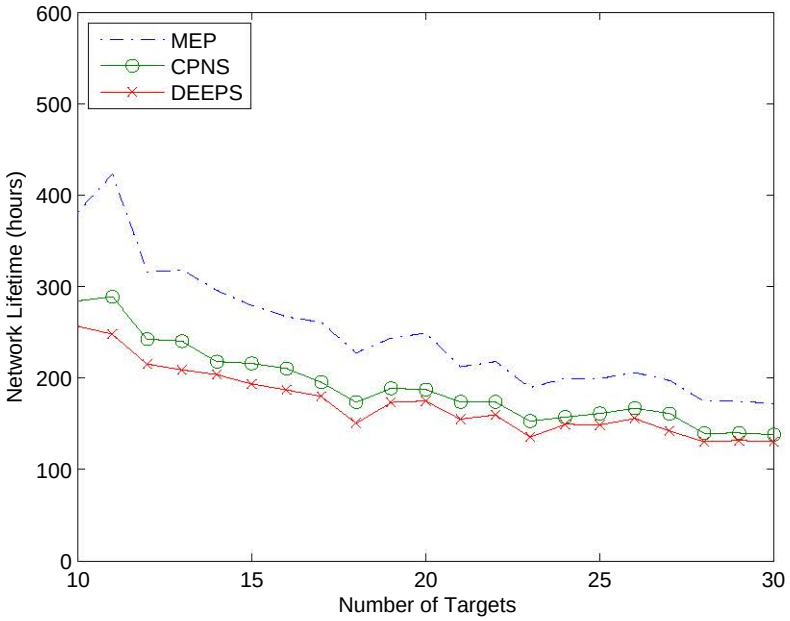
- *Network lifetime*. This is the time duration from which a network starts operation to when a target is not watched by any sensor nodes.
- *Average redundancy*. This is the average redundancy for each target. It represents the average number of overlapping sensor nodes for each target. Any on-duty sensor nodes monitoring a target covered by other on-duty nodes will increase the average redundancy.

5.3.1 Results

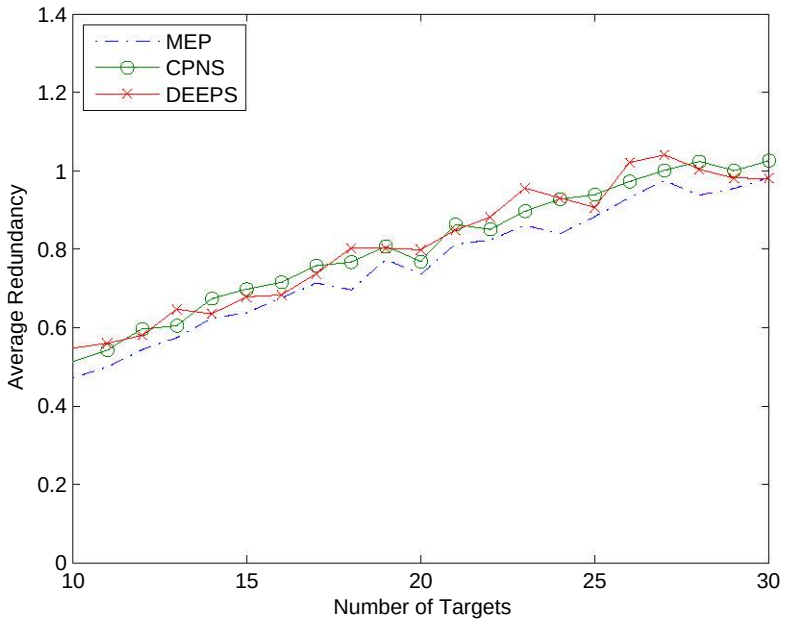
5.3.1.1 Target Density

The first experiment studies the impact of target density on the lifetime and average redundancy of MEP, CPNS and DEEPS. Assume that ten sensor nodes are deployed within a $1000 \times 1000m^2$ field, each has a uniform sensing range of 500 meters. The number of targets is increased from ten to 30.

From Figure 5.1a, it can be seen that the network lifetime decreases when there



(a)



(b)

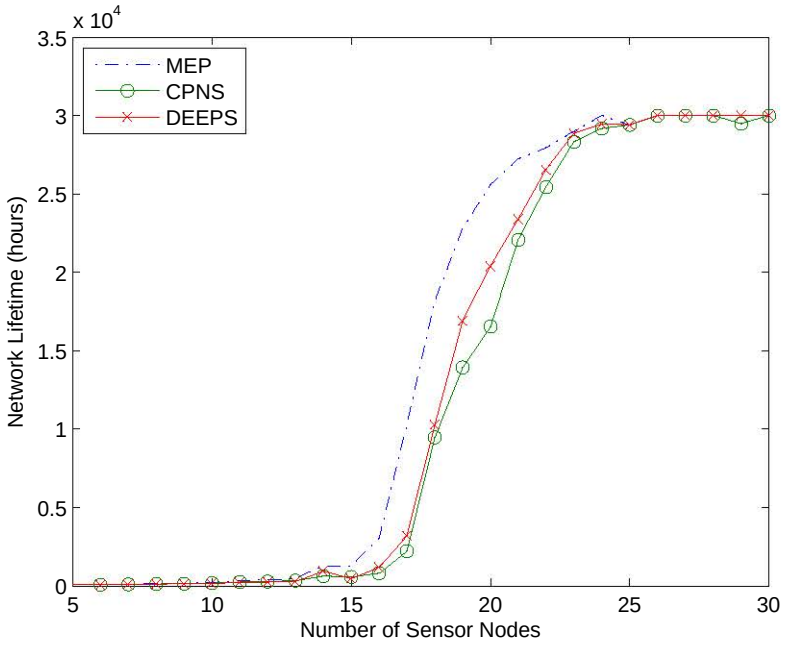
Figure 5.1: (a) Network lifetime and (b) average redundancy under different target densities

are more targets. The reason is that as the number of targets increases from ten to 30, each sensor node covers more targets within its sensing range. As a result, it has fewer opportunities to enter the sleep state to recharge itself, which reduces network lifetime. Additionally, in Figure 5.1a, it shows that the network lifetime of MEP is significantly higher than CPNS and DEEPS; i.e., 30% longer. The reason is that less energy is wasted when using MEP due to loss energy harvesting opportunities. Figure 5.1b shows that the average redundancy increases as the number of targets increases. This is because more targets are in the overlapping region of multiple sensor nodes. Moreover, MEP has 0.25 less average redundancy as compared to CPNS and DEEPS. This is due to MEP's eligibility test, which helps reduce redundancy. On the other hand, the redundancy of CPNS and DEEPS remains until next round or an overlapped sensor node exhaust its energy.

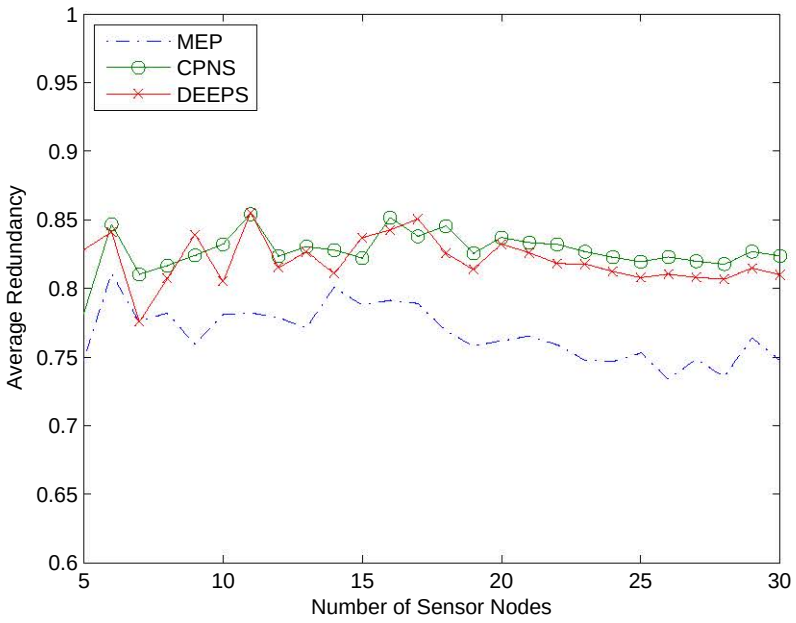
5.3.1.2 Node Density

In this experiment, the number of targets is fixed to 20 and the number of sensor nodes is varied from five to 30 - both sensor nodes and targets are dispersed within a $1000 \times 1000m^2$ field. All sensor nodes also have a uniform sensing range of 500 meters.

Figure 5.2a shows that when the number of sensor nodes increases from 15 to 25, the network lifetime of MEP, CPNS and DEEPS rapidly increases from less than 1000 hours to perpetual operation. The reason is that sensor nodes have more opportunities to be in the sleep state, which increase harvested energy. In this experiment, MEP outperforms CPNS and DEEPS. In particular, MEP achieves perpetual operation with one less sensor node than CPNS and DEEPS. Figure 5.2b shows that the average redundancy of MEP is less than CPNS and DEEPS. Moreover, the average redundancy of MEP reduced from 0.8 to 0.75 as the number of sensor nodes increase while the redundancy of CPNS and DEEPS remain at 0.82. That is, MEP achieves 30% longer lifetime and 10% lower redundancy than CPNS and DEEPS.



(a)



(b)

Figure 5.2: (a) Network lifetime, and (b) average redundancy under different sensor node densities.

5.3.1.3 Sensing Range

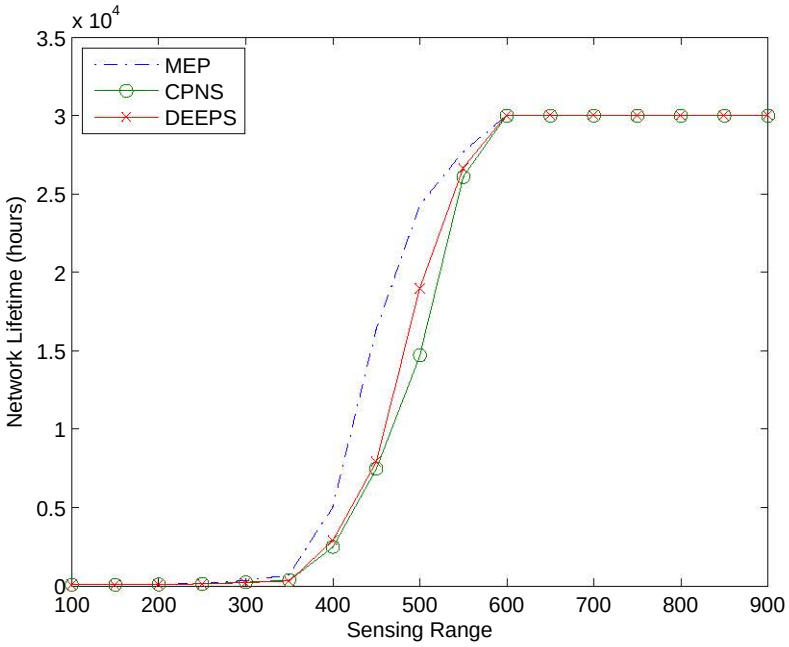
This experiment studies the effect of sensing range on network lifetime by varying the sensor nodes' sensing range from 100 to 900 meters. Assume that there are 20 sensor nodes and targets are dispersed within a $1000m^2$ field.

From Figure 5.3a, it can be seen that the network lifetime of all these algorithms starts rising when the sensing range is 350 meters. Perpetual operation is achieved when the sensing range equals 600 meters. However, CPNS and DEEPS require 50 meters more sensing range than MEP to achieve the same network lifetime. For example, the network running MEP can operate for 15000 hours when sensing range is 410 meters. To achieve such a network lifetime, CPNS and DEEP require a sensing range of 460 and 480 meters respectively. This is because there are more targets are redundant covered when using CPNS and DEEPS.

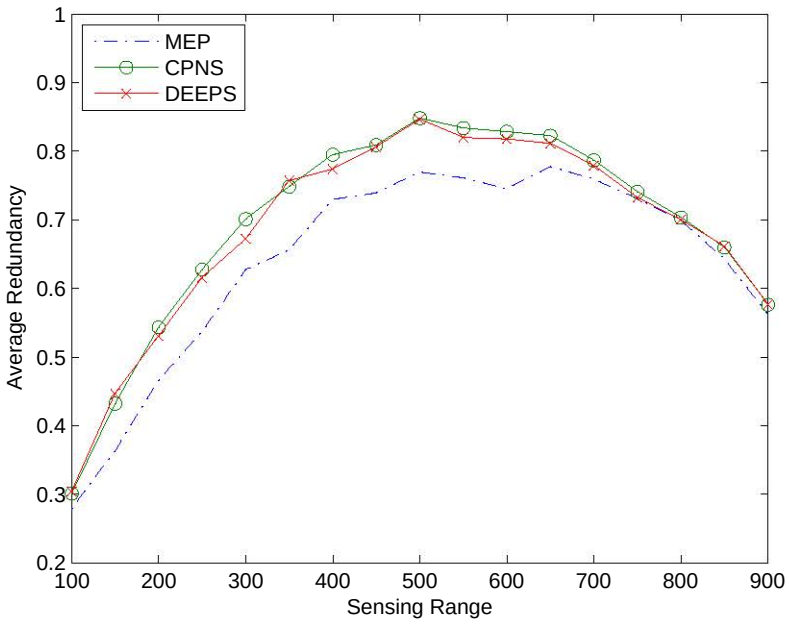
Figure 5.3b shows that the average redundancy peaked when the sensing range is 500 meters. The reason is that, when sensing range is small, each sensor node only covers a small number of targets and overlaps less with other nodes. When the sensing range is large, each sensor node will cover almost all targets, meaning not many sensor nodes are required to maintain coverage. This helps reduce redundancy.

5.4 Conclusion

This chapter has presented the first solution that addresses the MLCEH problem in a distributed manner. Sensor nodes decide by themselves whether to enter the active or sleep state based on the information from its two hop neighbors. The distributed algorithm called MEP allows only a subset of sensor nodes to enter the active status to maintain complete targets coverage. MEP also optimizes the energy harvesting opportunities of sensor nodes. MEP has been compared against two distributed algorithms designed for conventional non-rechargeable WSNs. The results show that MEP increases network lifetime by at least 30% and reduces redundancy by 10%.



(a)



(b)

Figure 5.3: (a) Network lifetime, and (b) average redundancy under different sensing ranges.

A critical assumption made by the solutions in Chapter 3, 4 and 5 is that they assume sensor nodes are able to connect to the base station/sink directly. In practice, this is not valid because sensor nodes only have a limited communication range. In other words, they need multi-hop communications. Henceforth, the next chapter considers connectivity and flows when addressing the MLCEH problem.

Chapter 6

Duty Cycling for Complete Targets Coverage and Connectivity

This chapter considers the problem of maximizing network lifetime whilst ensuring complete targets coverage and connectivity; also called the MLCCEH problem. The objective is to determine subsets of sensor nodes that have connectivity to the sink, and their corresponding active time. The key difference to the previous chapter is the additional requirement whereby all active sensor nodes must have at least one path to the base station/sink. Moreover, the network lifetime is now also dependent on when a sensor node is disconnected from the sink.

To date, as highlighted in Chapter 2, past works have only addressed the complete targets coverage and network connectivity problem in non-rechargeable WSNs. However, there are only a handful of works in the context of energy harvesting WSNs, e.g., [76][78], that have considered the complete targets coverage problem. Critically, these works do not consider network connectivity to the sink; i.e., scheduling the wake-up time of sensor nodes in order to forward sensed data back to the sink.

Henceforth, this chapter considers the following approaches. The first uses an exhaustive search [37] to generate a collection of set covers that provide coverage and

connectivity. These set covers are then fed into a LP solver to derive the minimal activation time of each set cover. However, the computation time of this approach increases exponentially with the number of sensor nodes. The second approach is an efficient energy conservation heuristic; aka EC-MLCCEH. It iteratively selects sensor nodes to monitor targets and ensure network connectivity based on their residual energy. Simulation results show that EC-MLCCEH achieves 80% of the network lifetime obtained via the first approach using only a fraction of the computation time.

6.1 Network Model

We will need two bipartite graphs: $G(S \cup Z, E)$ and $G'(S, N \cup \mathcal{N})$. In graph $G(S \cup Z, E)$, the sets S and Z contain sensor nodes and targets, respectively. The set of edges that connect a sensor $s_i \in S$ to one or more targets in Z is denoted by E. As for graph $G'(S, N \cup \mathcal{N})$, N is the set of edges connecting a sensor node $s_i \in S$ to one or more sensor nodes within its *communication* range. On the other hand, the set $\mathcal{N}$ contains edges that represent sensor nodes that are in each other's *sensing* range.

Note, as before, s_i and z_j index sensors and targets, where $i = 1 \ldots |S|$ and $j = 1 \ldots |Z|$. Let $Z(s_i)$ and $S(z_j)$ be a function that returns the set of targets covered by sensor s_i and the set of sensors covering target z_j respectively. With a slight abuse of notation, let $N(s_i)$ and $\mathcal{N}(s_i)$ return the set of sensor nodes within the communication and sensing range of sensor node s_i, respectively. Let B be the sink. Without loss of generality, assume all sensor nodes generate R bit/s when they monitor a target. Let $f_{ih} > 0$ be the data flow rate from sensor node s_i to s_h. All sensor nodes in S have at least one path to forward data to B when all sensor nodes are in the active state.

Assume time is discrete. The length of each time interval is δ_t, and is indexed by $t = 1, \ldots, T$. Let δ_T be the last time interval when a target is not monitored by a

sensor node, or when a sensor node does not have a route to B. Thus, the network lifetime is equal to $\sum_{t=1}^{T} \delta_t$.

Define $C_k \subseteq S$ to be a subset of sensor nodes, where $k = 1 \dots K$. Let $\phi(C_k, s_i)$ be a function that returns one if sensor node s_i is in the subset C_k, otherwise it returns zero. Each subset C_k has an activation time c_k^t in time interval t. Thus, the *total* activation time for sensor node s_i in time interval t is $x_i^t = \sum_{k=1}^{K} \phi(C_k, s_i) c_k^t$. Let $E_i^t \leq E_{max}$ be the battery level of sensor node s_i at the beginning of the time interval t. Specifically, it is equal to the battery level of sensor node s_i minus its consumed energy plus harvested energy in time interval $t-1$. Mathematically, E_i^t is calculated as follows:

$$E_i^t = E_i^{(t-1)} - (E^a x_i^{(t-1)} + E^{cr} f_{hi} + E^{ct} f_{ih}) + E_i^r \delta_{t-1} \tag{6.1}$$

where E_i^r (Joule/s) is the energy harvesting rate of sensor node s_i. The energy consumption rate of a sensor node when active is denoted by E^a (Joule/s). When it transmits or receive one bit, the energy incurred is denoted by E^{cr} (Joule) and E^{ct} (Joule), respectively. The energy consumption rate due to forwarding is denoted as E^f (Joule/bit); i.e., $E^f = E^{cr} + E^{ct}$. Lastly, we assume all sensor nodes have a full battery E_{max} at $t = 0$.

6.2 Solution

We are now ready to present two solutions for the MLCCEH problem. The first, called LP-MLCCEH, yields the optimal solution using an LP solver. The second is a heuristic, called EC-MLCCEH, where sensor nodes are selected based on their residual energy level.

6.2.1 LP-MLCCEH

The objective of LP-MLCCEH is to maximize network lifetime; i.e., determine the biggest T such that $\sum_{t=1}^{T} \delta_t$ is maximized. Here, all time intervals are assumed to have the same length. The main idea is to minimize the activation time of sensor nodes in each time interval whilst maintaining complete targets coverage and network connectivity. This ensures sensor nodes have ample time to conserve and recharge their battery.

Given the bipartite graphs $G(S \cup Z, E)$ and $G'(S, N \cup \mathcal{N})$, LP-MLCCEH first uses an exhaustive search to determine all possible subsets of sensor nodes C_k to cover all targets in Z and these sensor nodes must have a path to the sink. The resulting subsets constitute a collection of set covers $\Omega = \{C_k \mid k = 1 \dots K\}$.

After that, LP-MLCCEH aims to minimize $\sum_{i=1}^{|S|} x_i^t$ in each time interval t by determining the activation time of a subset of covers $\Gamma \subseteq \Omega$ such that their energy constraint is not violated. Using $x_i^t = \sum_{k=1}^{K} \phi(C_k, s_i) c_k^t$, we have the following LP,

$$\text{MIN} \sum_{t=1}^{T} \sum_{i=1}^{|S|} x_i^t \tag{6.2}$$

Subject to:

$$\sum_{k=1}^{K} c_k^t \geq 1 \quad , \quad \forall t = 1, \dots, T \tag{6.3}$$

$$0 \leq x_i^t \leq 1, \quad \forall i \in S, \quad \forall t = 1, \dots, T \tag{6.4}$$

$$R \sum_{j \in Z(s_i)} x_i^t + \sum_{h \in N(s_i)} f_{hi}^t = \sum_{h \in N(s_i)} f_{ih}^t + f_{iB}^t, \quad \forall i \in S, \quad \forall t = 1, \dots, T \tag{6.5}$$

$$\sum_{i \in S} f_{iB}^t = R \sum_{i \in S} \sum_{j \in Z(s_i)} x_i^t, \quad \forall t = 1, \dots, T \tag{6.6}$$

$$E_{max} - \sum_{t=1}^{T} \left[E^a x_i^t + E_i^{cr} f_{hi}^t + E_i^{ct} f_{ih}^t - E_i^r \right] \geq 0, \quad \forall s_i \in S \tag{6.7}$$

Constraint (6.3) ensures all targets are continuously covered in any time interval t.

Constraint (6.4) ensures each sensor node cannot be activated for more than one unit time in a time interval. Constraints (6.5) and (6.6) ensure flow conservation. Note, the data flow f_{iB} will be zero if node s_i is not directly connected to the sink. Constraint (6.7) ensures each sensor node does not spend more than its available energy.

To determine the best T value, binary search is applied repeatedly on the presented LP, which contains $T \times |\Omega| \times |S|^2$ decision variables and $T \times (|S| + |S| \times \Omega) + 2|S|$ constraints. In practice, the exhaustive search for all set covers Ω is computationally intensive [37]. Moreover, LP-MLCCEH incurs a high computational cost due to multiple calls to a LP solver for each T value. To this end, the next section presents an efficient heuristic solution for the MLCCEH problem.

6.2.2 Energy Conservation Heuristic (EC-MLCCEH)

EC-MLCCEH runs at the start of each time interval to determine its length and the set of sensor nodes to be activated. The main idea is to ensure sensor nodes with a higher residual battery level are used for sensing and relaying. EC-MLCCEH contains three main steps. It first determines the set of sensor node C_g to cover all the targets. It then chooses relay nodes to forward sensed data from nodes in C_g to the sink. At this point, these relay nodes as well as the nodes in C_g form a set C_t, which is to be activated in time interval t to ensure complete targets coverage and connectivity.

The next problem is to determine the activation time of C_t, denoted as δ_t. EC-MLCCEH sets the activation time of C_t to the minimum of the following values: (i) the minimal battery lifetime of sensor nodes in C_t, or (ii) the minimal recharging time of sensor nodes not in C_t. This means whenever a sensor node's battery is exhausted or at capacity, EC-MLCCEH generates a new subset of sensor nodes to be activated. This is an important consideration because sensor nodes are not able to store harvested energy when their battery is at capacity, and thus, EC-MLCCEH

minimizes the energy wastage due to lost recharging opportunities.

Before showing how EC-MLCCEH constructs C_g and C_t, a few key definitions are in order. Given the bipartite graph $G(S \cup Z, E)$, each sensor node $s_i \in S$ has up to K paths, which can be computed using Yen's algorithm [79]. Let P_i^k indicate the k-th path from sensor node s_i to B. All paths are stored in a collection $\Psi = \{P_i^k \mid \forall i \in S, k = 1, \ldots, K\}$. The function $S(P_i^k)$ returns the set of sensor nodes on path P_i^k excluding the source node, and $I(P_i^k)$ returns the set of indices belonging to sensor nodes returned by $S(P_i^k)$. The function $S(P_i^k, s_h)$ returns the set of sensor nodes in $S(P_i^k)$ that forward data through node s_h, where $s_h \in S(P_i^k)$. As an example, consider Figure 6.1. We have the path s_3-s_2-s_1-B. In this case, $S(P_3^1) = \{s_2, s_1\}$; $I(P_3^1) = \{2, 1\}$; $S(P_3^1, s_1) = \{s_2\}$.

Let E_i^c denote the energy consumption rate of sensor node s_i. The value of E_i^c is initialized to the energy consumption rate incurred when s_i is in the active state; i.e., $E_i^c = E^a + E^{ct}R|Z(s_i)|$. Here, $|Z(s_i)|$ is zero if sensor node s_i does not monitor any targets. Sensor node s_i will have a higher energy expenditure if it is relaying for other sensor nodes. Let $F_h(P_i^k)$ and $D_h^c(P_i^k)$ denote the increased data forwarding rate and energy consumption rate of node s_h when s_i forwards its data via path P_i^k, respectively. As an example, consider Figure 6.1. Assume sensor node s_2 forwards its sensed data using the path s_2-s_1-B. Then E_1^c is increased by $D_1^c(P_2^1)$; i.e., $E_1^c = E_1^c + D_1^c(P_2^1)$. If s_3 forwards data using path $P_3^1 = s_3$-s_2-s_1-B, then $F_2(P_3^1) = F_1(P_3^1) = R$ and $D_2^c(P_3^1) = D_1^c(P_3^1) = E^f R$, where R is the data generated from target t_2. On the other hand, if s_3 forwards data using path $P_3^2 = s_3$-s_4-s_1-B, the increased data forwarding rate $F_4(P_3^2)$ is R whilst $F_1(P_3^2)$ is $2R$. In this case, the increased energy consumption rate of s_4 and s_1 is $E_4^c + E^f R$ and $2E^f R$, respectively. This is because the increase in energy consumption rate at s_4 is the sum of the energy consumption rate due to it being active as well as the energy cost of forwarding data from s_3. Moreover, activating s_4 results in additional data flowing through node s_1. Lastly, given the residual energy E_i^t of all sensor nodes $s_i \in S$ in time interval t, let $L(P_i^k)$ be the minimum operation time of the sensor nodes on the path P_i^k if s_i

forwards its data via path P_i^k; i.e., $L(P_i^k) = MIN\{\frac{E_h^t}{E_h^c + D_h^c(P_i^k)} \mid \forall h \in I(P_i^k)\}$.

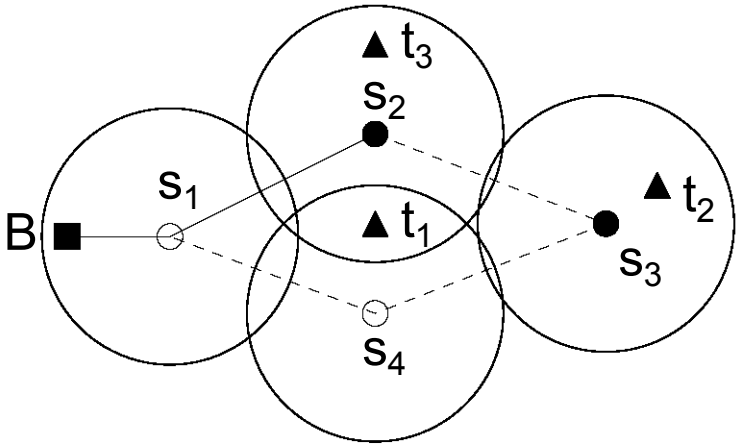


Figure 6.1: An example topology where filled and unfilled circles are sensor nodes monitoring or not monitoring targets respectively. Triangles are targets and the square is the sink.

6.2.2.1 Construction of C_g

Algorithm 3 shows how EC-MLCCEH constructs the set C_g. Given the residual energy E_i^t of all sensor nodes $s_i \in S$, EC-MLCCEH will carry out the following steps at the beginning of each time interval t. Let s_{i^*} be a sensor node in S with the minimal $E_{i^*}^t$ value; i.e., $i^* = \arg\min_{i \in S} E_i^t$. EC-MLCCEH will turn off s_{i^*} if all targets covered by s_{i^*} are monitored by other sensor nodes; see *Line* 5. This is because EC-MLCCEH ensures the sensor node with the lowest residual energy is turned off to conserve its remaining energy and provides it with opportunities to recharge. Critically, it checks that turning off s_{i^*} does not result in any targets being uncovered. EC-MLCCEH then removes s_{i^*} from S. If any target is only covered by s_{i^*}, sensor node s_{i^*} is removed from S and is added into the set C_g. This is because s_{i^*} needs to be activated to maintain complete targets coverage. EC-MLCCEH then repeatly tries to turn off the next sensor node with the minimal E_i^t until S is empty, see *Line* 2-13. If any target is not monitored, EC-MLCCEH exits and no set cover is available. Note, the sink B is considered to be active at all times.

Algorithm 3: Pseudocode for EC-MLCCEH (construction of C_g)

Input: $G(S \cup Z, E)$
Output: C_g

$C_g = \{B\}$
while $S \neq \emptyset$ **do**
  $i^* = \arg\min_{i \in S} E_i^t$
  Delete s_{i^*} from S
  if *all* $z_j \in Z(s_{i^*})$ *are covered* **then**
    Continue
  else
    Add s_{i^*} into C_g
    if $E_{i^*} == 0$ **then**
      Exit
    end
  end
end
if *any* $z_j \in Z$ *is not covered* **then**
  Exit
end

6.2.2.2 Construction of C_t

After EC-MLCCEH successfully generates the set C_g, it then proceeds to construct the set cover C_t; see Algorithm 4. EC-MLCCEH first adds all the sensor nodes in C_g into the set cover C_t. It then determines the path for the sensor nodes in C_g to forward their sensed data to the sink. Note that all the sensor nodes on a selected path will be added into the set cover C_t.

In order to choose a path with a longer operation time, for each sensor node s_i in the set C_g, EC-MLCCEH will calculate the minimum operation time of each s_i's path to the sink; i.e., $L(P_i^k)$. To do this, it first calculates the increased data flow rate $F_h(P_i^k)$ and energy consumption rate $D_h^c(P_i^k)$ when s_i forwards data through node s_h for all sensor nodes s_h on the path p_i^k. If any node s_h on the path P_i^k is not in the set C_t, i.e., $\delta(s_j, C_t) = 0$, then node s_h needs to be activated and added to the set C_t. Thus, the energy consumption rate of all the sensor nodes on the path from s_h to the sink will be increased according to the sensed data generated by s_h. Consequently, the increased data flow and energy consumption rate of s_h is

calculated as follows,

$$F_h(P_i^k) = \sum_{s_j \in S(P_i^k, s_h)} (1 - \delta(s_j, C_t))R|Z(s_j)| + R|Z(s_i)| \tag{6.8}$$

$$D_h^c(P_i^k) = (1 - \delta(s_h, C_t))E_h^c + E^f F_h(P_i^k) \tag{6.9}$$

The summation part of Equation (6.8) represents the increased data flow rate through s_h due to sensor nodes that are not in the set C_t. Note, the expression $(1 - \delta(s_h, C_t))E_h^c$ is zero if s_h belongs to the set C_t. Consider Figure 6.1. If s_3 forwards data through s_2, say path P_3^1, the increased energy consumption rate of s_1 is $E^f R$; i.e., $S(P_3^1, s_1) = \{s_2\}$ and $\delta(s_2, C_t) = \delta(s_1, C_t) = 1$ in Equation (6.8) and (6.9). However, if s_3 forwards data through s_4, then this rate at s_1 is $2E^f R$. This is because s_4 is not in C_t and $\delta(s_4, C_t) = 0$ in Equation (6.8).

EC-MLCCEH then selects a path $P_i^{k^*}$ that has the maximum $L(P_i^k)$ value to transmit the sensed data from s_i to the sink; i.e., $k^* = \arg\max_{k \in K} L(P_i^k)$; see *Line* 12. Thus, the sensor nodes with a low residual energy; i.e., the sensor nodes on the path P_i^k with a lower $L(P_i^k)$ value than $L(P_i^{k^*})$ are not responsible for forwarding data and their energy is conserved. If any sensor node s_h in $S(P_i^{k^*})$ is not in the set C_t, EC-MLCCEH will add s_h into the set C_t; see *Line* 13-18. After that, all sensor nodes in $S(P_i^{k^*})$ increment their energy consumption rate E_i^c by $D_h^c(P_i^{k^*})$. This procedure repeats for all sensor nodes in C_g, see *Line* 5-19. After obtaining C_t, EC-MLCCEH calculates the duration in which sensor nodes in C_t remain active. Let LO_t and LR_t be the minimal operation time of all sensor nodes in C_t and the minimal recharging time of all sensor nodes not in C_t respectively. Thus, the operation time δ_t of C_t is equal to MIN(LO_t, LR_t); see *Line* 20-22.

6.2.2.3 Analysis

The run time complexity of EC-MLCCEH is as follows.

Proposition 4. *EC-MLCCEH has a run time complexity of* $\mathcal{O}(|S|^2(\frac{1}{2}|Z| + K))$.

Algorithm 4: Pseudocode for EC-MLCCEH (construction of C_t)

Input: C_g, Ψ, E_i^t
Output: C_t and its activation duration δ_t

$C_t = C_g$
for *all* $s_i \in S$ **do**
  $E_i^c = E^a + E^{ct}R|Z(s_i)|$
end
for *all* $s_i \in C_g$ **do**
  for *all* $P_i^k \in P_i$ **do**
    for *all* $s_h \in P_i^k$ **do**
      Calculate $D_h^c(P_i^k)$ as per Equ. (6.8) and (6.9)
    end
    $L(P_i^k) = MIN\{\frac{E_h^t}{E_h^c + D_h^c(P_i^k)} \mid h \in I(P_i^k)\}$
  end
  $k^* = \arg\max_{k \in K} L(P_i^k)$
  for *all* $h \in I(P_i^{k^*})$ **do**
    $E_h^c = E_h^c + D_h(P_i^{k^*})$
    if $s_h \notin C_t$ **then**
      Add s_h into C_t
    end
  end
end
$LO_t = \text{MIN}\{\frac{E_i^t}{E_i^c} \mid s_i \in C_t\}$
$LR_t = \text{MIN}\{\frac{E_{max} - E_i^t}{E_i^r} \mid s_i \in S, s_i \notin C_t\}$
Return: C_t, $\delta_t = MIN(LO_t, LR_t)$

Proof. In Algorithm 3, the number of elements in S is reduced by one after each while iteration, see *Line* 4. Thus the total number of iterations is $\frac{(1+|S|)|S|}{2}$. In the worst case, each iteration is carried out for $\mathcal{O}(|Z|)$ times to search all targets. After that, *Line* 14-16 requires $\mathcal{O}(|Z|)$ steps for each time interval. Therefore, the running time complexity of Algorithm 3 is $\mathcal{O}(\frac{(1+|S|)|S|}{2}|Z| + |Z|)$. In the worst case, the number of sensor nodes in C_g is $|S|$, each has K paths to the sink and the number of sensor nodes on each path is $|S|$. Moreover, the procedure for determining the activation time of a calculated set cover, i.e., *Line* 20 and 21 in Algorithm 4, requires $\mathcal{O}(|S|)$ steps for each time interval. Thus, the running time complexity of Algorithm 4 is $\mathcal{O}(|S|(K|S| + |S| + 2))$ Therefore the total time complexity is $\mathcal{O}(\frac{(1+|S|)|S|}{2}|Z|+|Z|+|S|(K|S|+|S|+2))$ or $\mathcal{O}(|S|^2(\frac{1}{2}|Z|+K+1)+|S|(\frac{1}{2}|Z|+2)+|Z|)$, which is $\mathcal{O}(|S|^2(\frac{1}{2}|Z| + K))$. □

6.3 Evaluation

The experiments are conducted in Matlab running on an Intel Core i7 CPU @ 3.5GHz with 8 G RAM computer. The sensor node parameters correspond to the WaspMote [72] platform. A sensor node consumes 60 mW when in the active state and 0.2 mW when sleeping. Moreover, all sensor nodes are equipped with an Enocean ECS310 solar cell [73] to harvest solar energy. This solar cell is assumed to have a conversion rate of 10% and a recharging efficiency of 50%, which is conservative as compared to other technologies [74]. In addition, all experiments use real solar irradiance data retrieved from a sunny day at Southwest Solar Research Park, Phoenix, Arizona, USA [75] on the 16-th of April 2013. The average recharging rate is derived from the data collected in 24 hours. Table 6.1 shows the parameter values used in all experiments.

The experiments compare EC-MLCCEH with LP-MLCCEH. In the EC-MLCCEH, it calculates three shortest paths from each sensor node to the sink, i.e., $K = 3$. The experiments also compare EC-MLCCEH and LP-MLCCEH to a heuristic pro-

Parameters	Value
Battery size	1100 mA
Consumption rate	3.6 Joules/minute
Data generation rate	3.8 Kbytes/minute
Average recharge rate	0.96 Joules/minute
Voltage	4 V
Solar panel conversion rate	10 %
Recharging efficiency	50 %
Transmission cost	0.1 Joules/Kbyte

Table 6.1: Simulation Parameters

posed in [52] called Random Activation for MLCCEH (RA-MLCCEH). For the RA-MLCCEH, all sensor nodes have an identical probability to be activated at the beginning of each time interval t to construct a subset C'_t. If C'_t does not guarantee complete targets coverage and network connectivity, then it discards this subset and repeats the process until one is found. RA-MLCCEH will terminate if no subsets are found when a predetermined iteration limit is reached. For a C'_t that guarantees coverage and connectivity, define its operation time to be $MIN(LO_t, LR_t)$. This is because in energy harvesting WSNs, RA-MLCCEH needs to consider recharging opportunities of sensor nodes.

The experiments investigate the *network lifetime* and *running time* by varying the following parameters: sensor node density, target density and sensing range. Here, *network lifetime* is the time duration from which a network starts operation to when any target is not watched by any sensor nodes or a sensor node is not connected to the sink. The experiments assume sensor nodes and targets are dispersed within a $100 \times 100\ m^2$ field. All sensor nodes have the same sensing range and their communication is twice the sensing range. Each experiment is conducted with one change to the network configuration while others are fixed. Details of the configurations will be made specific in Section 6.3.1. The results are an average of 100 runs, each with a different topology.

6.3.1 Network Lifetime

The following sections show the results on coverage lifetime when the number of sensor nodes, targets and sensing range changes.

6.3.1.1 Sensor Node Density

In the first experiment, the number of targets is fixed to five and the number of sensor nodes is varying from two to ten with an interval of one. All sensor nodes have an uniform sensing radius of 30 meters. From Figure 6.2, it can be seen that the network lifetime of LP-MLCCEH and EC-MLCCEH increases rapidly when adding more sensor nodes. This is because each sensor node has more opportunities to enter sleep state to recharge as the number of sensor nodes increases. In the RA-MLCCEH algorithm, randomized scheduling of sensor nodes lead to unnecessary activation, e.g., a target is monitored by multiple sensor nodes, and hence result in energy waste. Figure 6.2 also shows that EC-MLCCEH achieves 80% performance in terms of network lifetime as compared to LP-MLCCEH. This is because EC-MLCCEH minimizes the number of sensor nodes to be activated in each time interval, see Algorithm 3. This helps prolongs network lifetime.

6.3.1.2 Target Density

In this experiment, the number of sensor nodes is fixed to eight and the number of targets varies from two to 10 with an interval of one. The sensing range of all sensor nodes is 30 meters. Figure 6.3 shows the network lifetime of LP-MLCCEH and EC-MLCCEH rapidly reduces from 1800 and 1100 hours to less than 200 hours with increasing number of targets from two to five. The reason is because each sensor node needs to have more activation time to monitor the newly added targets. Thus they have fewer opportunities to enter the sleep state to recharge themselves, which reduces network lifetime.

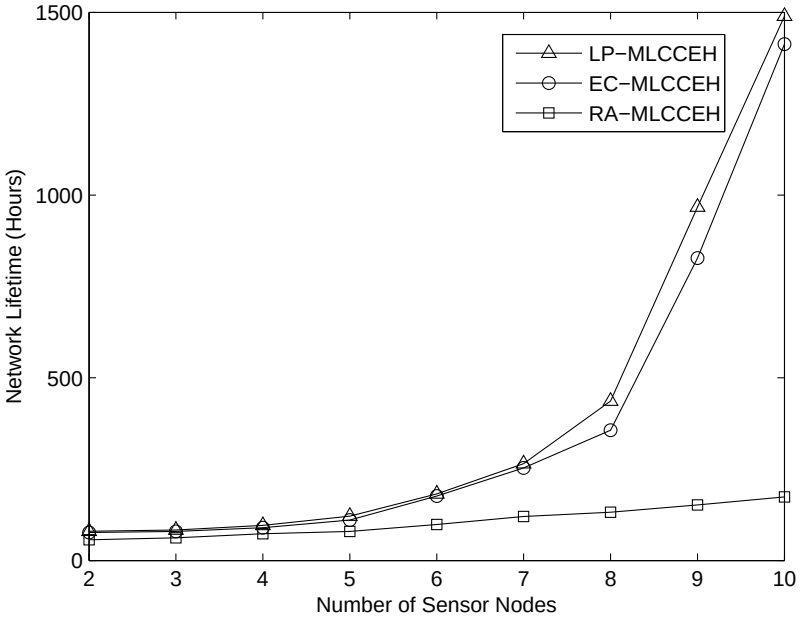


Figure 6.2: Network lifetime when varying the sensor node densities

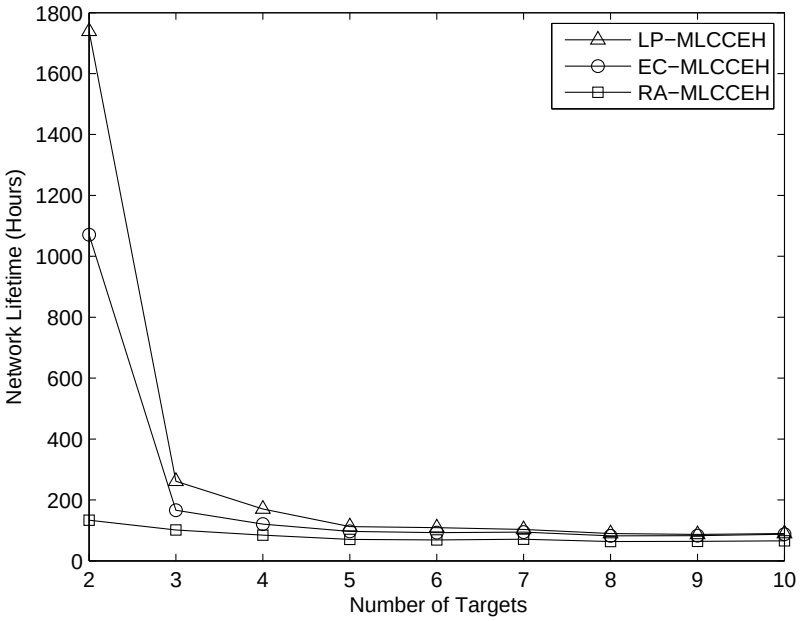


Figure 6.3: Network lifetime when varying the target densities

6.3.1.3 Sensing Range

In the last experiment, there are eight sensor nodes and five targets. However, the sensing radius is varied from five to 50 meters with an interval of five. From Figure 6.4, it shows that the network lifetime of LP-MLCCEH, EC-MLCCEH and RA-MLCCEH increases with the increasing of sensing range. This is because each sensor node is able to cover more targets, and thus, the total number of sensor nodes to be activated is reduced. As a result, sensor nodes have more opportunities to be in the sleep state.

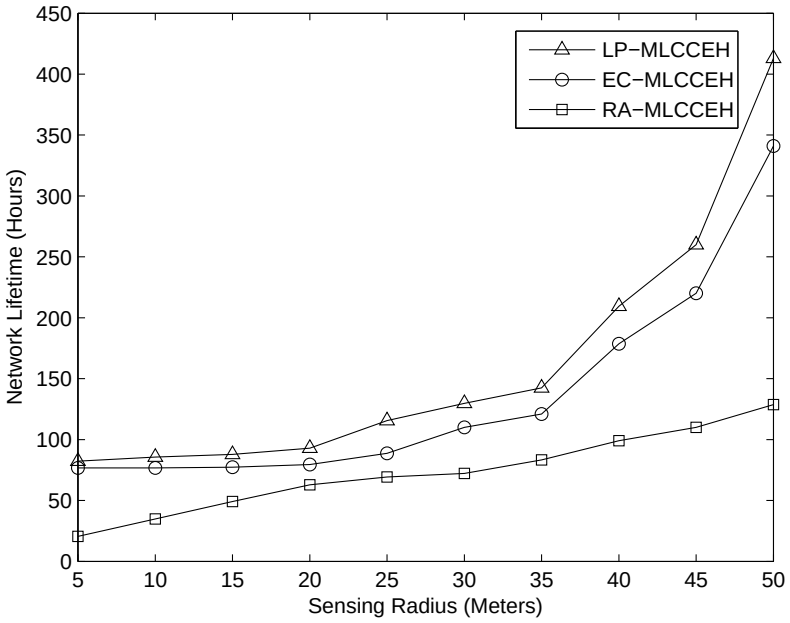


Figure 6.4: Network lifetime when varying the sensing range

In comparison to the three experiments, it shows that the performance of EC-MLCCEH is close to LP-MLCCEH in terms of network lifetime. On the other hand, the network lifetime of RA-MLCCEH is less than 200 hours in all three experiments, which is significantly lower than that of EC-MLCCEH.

6.3.2 Running Time

Figure 6.5, 6.6 and 6.7 show the running time of LP-MLCCEH, EC-MLCCEH and RA-MLCCEH when varying the number of sensor nodes, targets and sensing range,

respectively. From Figure 6.5 and 6.7, it can be seen that the running time of LP-MLCCEH increases exponentially with increasing number of sensor nodes and sensing range. This is because the number of set covers, i.e., Ω, increases with more sensor nodes or a larger sensing range. Thus, the number of decision variables to be decided by the LP solver increases. Moreover, with increasing of network lifetime T, there will be more calls to the LP solver. On the other hand, EC-MLCCEH iteratively selects sensor nodes at each time interval, which incurs significantly less computational cost. RA-MLCCEH has the lowest operation time when increasing the number of sensor nodes and targets, see Figure 6.5 and 6.6. This is because sensor nodes waste energy due to improper activation schedule, which result in low network lifetimes and incurs fewer iterations to generate set covers. However, Figure 6.7 shows the running time of RA-MLCCEH is higher than LP-MLCCEH when sensing range is less than 25 meters. The reason is because RA-MLCCEH needs a longer time to construct the set covers that ensure coverage and connectivity when sensor nodes have a small sensing range.

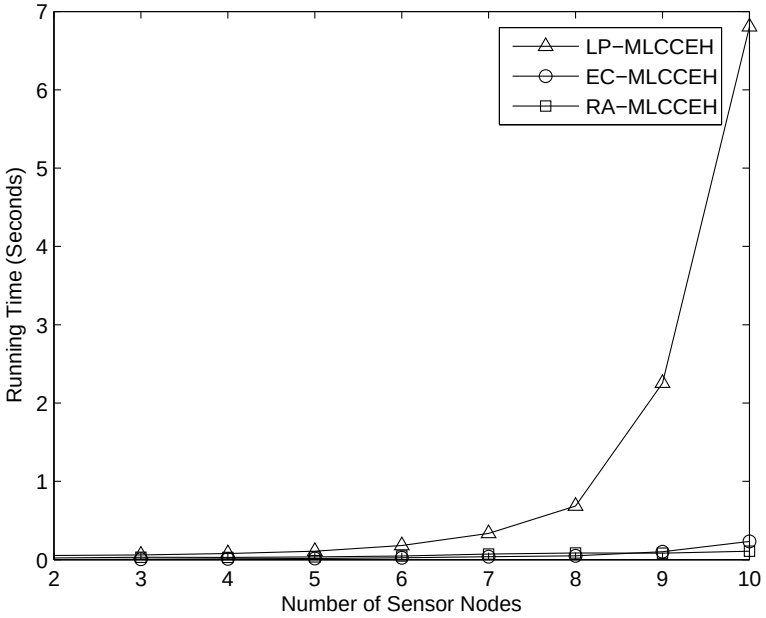


Figure 6.5: Network lifetime when varying the sensor node densities

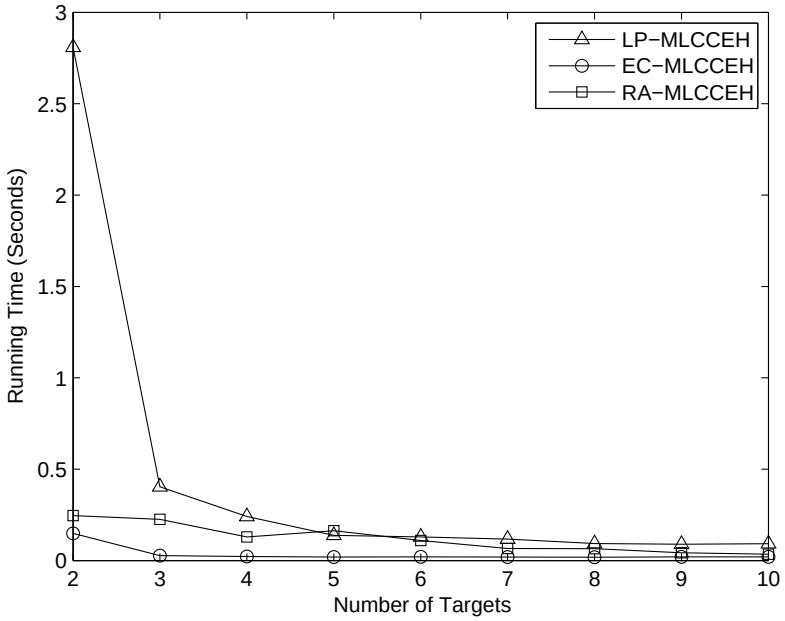


Figure 6.6: Network lifetime when varying the target densities

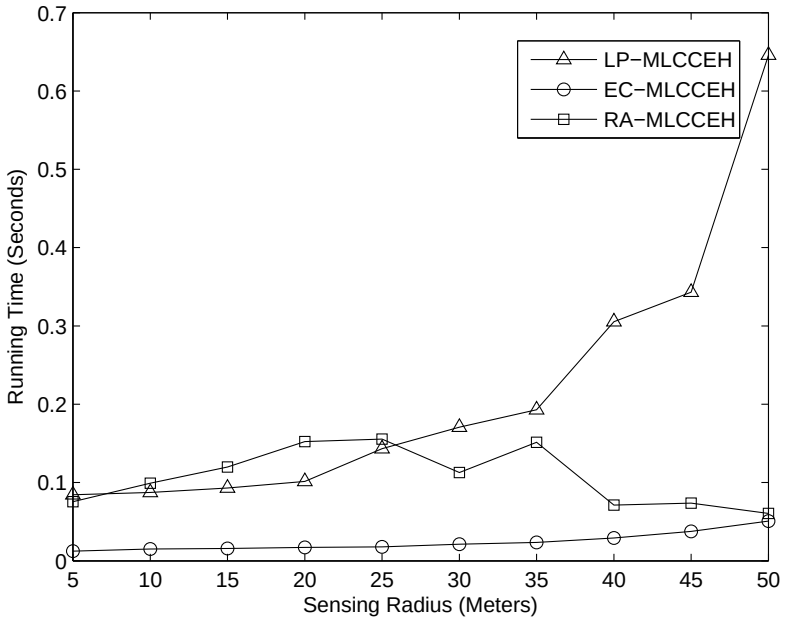


Figure 6.7: Network lifetime when varying the sensing ranges

6.4 Conclusion

This chapter has presented the first solutions to address the network connectivity problem along with the complete targets coverage problem in energy harvesting WSNs. The proposed solutions ensure the set of sensor nodes that monitor targets are connected to the sink. Critically, these solutions ensure sensor nodes are afforded recharging opportunities. In particular, the LP-MLCCEH algorithm formulates the said problem as a LP based on an exhaustive collection of set covers. Then, a heuristic algorithm, called EC-MLCCEH, selects sensor nodes into a set cover based on their energy level. Simulation results show that the EC-MLCCEH achieves 80% of the network lifetime attained by the LP-MLCCEH.

A key assumption of the duty cycle scheduling algorithms studied in Chapter 3, 4, 5 and 6, is that they assume all sensor nodes are already deployed around targets. The next chapter outlines solutions that relax this assumption and consider the placement of energy harvesting sensor nodes.

Chapter 7

On Node Placement Problem for Energy Neutral Operation

As highlighted in Chapter 1 and 2, past duty cycling algorithms assume sensor nodes are already deployed around targets. Moreover, in terms of nodes placement for complete targets coverage problem, existing works assume nodes have no ability to replenish their energy storage. Their main objective is to minimize the number of sensor nodes used to cover all targets. Alternatively, they may require targets to be monitored by k sensor nodes [37] for reliability reasons. However, their network lifetime is restricted by the battery capacity of sensor nodes. The work in this chapter is different in that its context is energy harvesting WSNs whereby it focuses on node deployment/placement to achieve energy neutral target coverage. This is achieved when the energy consumption rate is less than or equal to the total rate in which energy is harvested by sensor nodes monitoring targets. That is, unlike prior works, this chapter considers sensor nodes with energy harvesting capability and varying energy harvesting rates afforded by different locations. This new problem is called MEHNP-PC: given a sensing field divided into cells, each of which has an i.i.d energy harvesting rate, determine the cells and the minimum number of sensor

nodes to be placed in these cells such that all *fixed* targets are covered perpetually.

In this chapter, the MEHNP-PC problem is formulated as an ILP, which is NP-hard in general. The objective is to minimize the total number of sensor nodes used to cover all targets whilst ensuring energy neutral operation. In fact, the NP-complete minimum set cover problem is a special case of MEHNP-PC problem. After that, a number of relaxations of the said ILP are proposed and investigated. In particular, three algorithms are proposed, namely GRNP, TPNP and EENP, to round fractional solutions. Experimental results show that EENP outperforms GRNP and TPNP, and is close to optimal with a 1% gap in terms of the required number of sensor nodes.

7.1 Network Model

In this chapter, the sensing area with targets is modeled as a bipartite graph $G(C, Z, E, W)$, where C is the set of cells whilst Z is a set of *fixed* targets. The set E contains edges that indicate whether a cell can be used to monitor a target; e.g., an edge (u, v), where $u \in C$ and $v \in Z$, indicates a cell c can be used to monitor target v. The set W contains edge weights, which will be elaborated in the next paragraph. Assumes a WSN is deployed on a square sensing field that is divided into $L = l \times l$ grid cells. Each cell c has coordinate $c_{(m,n)}$, where $m = 1 \dots l, n = 1 \dots l$. In addition, each cell has an i.i.d recharging rate $R_{(m,n)}$ (Watts). Let S be the set of sensor nodes. The notations s_i and z_j are used to index sensor nodes and targets, where $i = 1 \dots |S|$ and $j = 1 \dots |Z|$. Let E_i^r (Watts) be the recharging rate of sensor node s_i, which is equal to $R_{(m,n)}$ if s_i is placed in cell $c_{(m,n)}$. Each sensor node has a battery capacity of B_{max} (Joule) and has an energy consumption rate of E^c (Watts) when active. The energy consumed during sleep is assumed negligible. Moreover, targets are assumed stationary and randomly placed at the center of cells. Each cell contains at most one target. However, there is no limit on the number of sensor nodes in each cell. Sensor nodes located in the same cell have an identical recharging

rate.

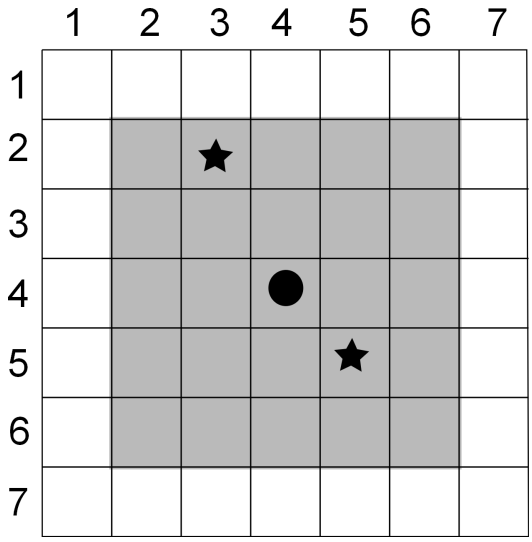


Figure 7.1: Sensing range when $u = 3$, where a sensor node is denoted by a solid circle.

Let $Z(S_i)$ be a function that returns the set of targets covered by sensor node s_i. Conversely, $S(z_j)$ is a function that returns the set of sensor nodes covering target z_j. With a slight abuse of notation, let $Z(c_{(m,n)})$ return the set of targets covered by cell $c_{(m,n)}$; i.e., if a sensor node s_i is placed in cell $c_{(m,n)}$ then it will cover $|Z(s_i)|$ targets. Let $C(z_j)$ return the set of cells covering target z_j. Defines the weight W of a cell to be $w_{(m,n)} = R_{(m,n)} \times |Z(c_{(m,n)})|$. That is, a cell's weight is proportional to its recharging rate and the number of targets it can cover.

Assumes all sensor nodes have the same sensing radius u. The sensing range of a sensor node is denoted as γ. Without loss of generality and for ease of exposition, the sensing range is assumed to be a square area. For example, when the sensing radius is two, a node can monitor nine cells, which includes the cell it is located in and eight other cells surrounding the node. Each sensor node is able to communicate with the base station or sink directly. Hence, the connectivity problem, see [80], which calls for a solution that ensures a connected network is beyond the scope of this problem. It is remarkable that as shown in [80], if the communication radius is twice the sensing radius, the resulting network is connected when the entire sensing field is completely covered by sensor nodes. Assume sensor nodes are able to sense

omni-directionally and thus are capable of monitoring one or more targets with equal energy consumption rate. In Figure 7.1, it shows a sensor node s_i located in cell $c_{(4,4)}$ and two targets in cell $c_{(2,3)}$ and $c_{(5,5)}$, respectively. When sensing radius is three cells, the cells covered by s_i are indicated by the gray cells. Also, sensor s_i will consume the same energy to monitor any number of targets placed in these cells. Note that a sensor node only covers the cell it is located in if its sensing radius is one.

7.2 Problem Statement

In this chapter, the aim is to determine the minimum number of sensor nodes and their location such that all targets are covered whilst ensuring energy neutral operation/perpetual coverage. In this new problem, termed MEHNP-PC, each cell has a different recharging rate. It is remarkable that the problem of scheduling the active or sleep state of sensor nodes, i.e., after they are placed in their cell, is complementary to this work. Interested readers are referred to [76].

A WSN has *total perpetual coverage* or energy neutral operation when the total harvesting rate of sensor nodes monitoring each target is a higher than the energy consumption rate attributed to monitoring [25]. Formally, total perpetual coverage is achieved when $\sum_{i \in S(z_j)} E_i^r \geq E^c$ for all targets $z_j \in Z$. The problem at hand can be modeled as an ILP formulation. Let $x_{(m,n)}$ be the number of sensor nodes placed in cell $c_{(m,n)}$. Recall that the energy harvesting rate in cell $c_{(m,n)}$ is $R_{(m,n)}$. Then we have the following formulation.

$$\text{MIN} \sum_{(m,n) \in C} x_{(m,n)} \tag{7.1}$$

subject to

$$\sum_{(m,n)\in C(z_j)} R_{(m,n)} \times x_{(m,n)} \geq E^c, \quad \forall z_j \in Z \tag{7.2}$$

$$x_{(m,n)} \in \mathbb{Z}^+ \quad \forall (m,n) \in C \tag{7.3}$$

Constraint (7.2) ensures total perpetual coverage; this is attained if the total recharging rate of sensor nodes covering each target exceeds E^c. Constraint (7.3) ensures the number of sensor nodes placed in each cell is a non-negative integer. Next proposition shows that there is no polynomial time algorithm to derive an exact solution for the MEHNP-PC problem.

Proposition 5. *The MEHNP-PC problem is NP-complete.*

Proof. A special instance of MEHNP-PC problem is the well known NP-complete, Minimum Set Cover Problem [81]. To see this, let the set of targets Z be the universe or ground set. Each cell or set contains all targets it covers should a node be placed in it; i.e., $Z(c_{(m,n)})$. Then set the recharging rate of each cell assume each cell to one Watt, and E^c is one Watt. As the MEHNP-PC problem is a generalization of the minimum set cover problem, it too cannot be solved in polynomial time. □

7.3 Solutions

The main idea is to relax the ILP presented in Section 7.2, whereby instead of restricting $x_{(m,n)}$ to be an integer, set it to be a real positive number. The revised formulation is thus an LP, which will be denoted as LP^*. In the subsequent sections, the optimal fractional solution from this LP is denoted as $x'_{(m,n)}$. Note that only the results that are strictly greater than zero are considered in $x'_{(m,n)}$.

A key problem in the relaxed LP is *rounding* $x'_{(m,n)}$ to an integer solution. In the subsequent sections, three rounding algorithms are proposed which have a provable approximation ratio. The first algorithm is called GRNP that rounds up any frac-

tional solutions. Although simple, it has a large approximation ratio. To this end, the second algorithm, called TPNP, rounds down instead. This, however, yields an infeasible solution, meaning it needs to construct a good feasible solution. Therefore, TPNP greedily places sensor nodes near targets that require the most energy. Unfortunately, the required number of sensor nodes computed can still be far from the optimal solution. So, EENP is proposed, a solution that also rounds down fractional solution, but unlike TPNP that considers one target at a time, EENP considers a group of targets. It is noticed that the only inputs required by the proposed algorithms are the available locations to place sensor nodes and their respective recharging rate. These can be obtained via a site survey, either manually or via autonomous vehicles.

The following subsections present the details of the proposed algorithms. Then Section 7.4 outlines proofs of the quality of the proposed solutions with respect to the optimal solution. This is followed by their run time complexity in Section 7.4.

7.3.1 Greedy Round Node Placement (GRNP)

GRNP simply *rounds up* all fractional $x'_{(m,n)}$ variables to their nearest integer. Consider a solution $x'_{(m,n)} = 1.2$; i.e., 1.2 sensor nodes are to be placed in cell $c_{(m,n)}$. Then GRNP will round up this number to two sensor nodes. In subsequent sections, let $\mathcal{G}_{(m,n)}$ to indicate the total number of sensor nodes, as determined by GRNP, to be placed in cell $c_{(m,n)}$. That is, $\mathcal{G}_{(m,n)} = \lceil x'_{(m,n)} \rceil$.

7.3.2 Target Protection Node Placement (TPNP)

TPNP first rounds down all fractional $x'_{(m,n)}$ value to its nearest integer; i.e., $\lfloor x'_{(m,n)} \rfloor$. Consequently, the revised solution is no longer feasible. TPNP then places sufficient sensor nodes around targets to satisfy constraint (7.2). Here, defines the *shortfall* of a target z_j, denoted by F_j, as the energy required for perpetual coverage minus the energy provided by the sensor nodes monitoring it after round down. Mathemati-

cally, the shortfall is calculated as follows:

$$F_j = E^c - \sum_{(m,n) \in C(z_j)} R_{(m,n)} \times \lfloor x_{(m,n)} \rfloor \tag{7.4}$$

The steps carried out by TPNP are illustrated in Algorithm 5, which operates on $x'_{(m,n)}$. Let $\mathcal{T}_{(m,n)}$ be the number of sensor nodes to be placed in cell $c_{(m,n)}$ calculated by TPNP. After rounding down $x'_{(m,n)}$, TPNP calculates the shortfall F_j of all targets based on Equation (7.4); see *Line* 3-5. It then determines the target z_{j^*} that has the maximum shortfall F_{j^*}; i.e., *Line* 9. TPNP then places $\lceil \frac{F_{j^*}}{R_{(m,n)^*}} \rceil$ sensor nodes into cell $c_{(m,n)^*}$, where $c_{(m,n)^*}$ is the cell that has the maximum recharging rate that can be used to monitor target z_{j^*}; see *Line* 10-11. After that, all targets update their F_j according to the newly placed sensor nodes; i.e., rerun the *Line* 3-5. TPNP then identifies the next target with the maximum shortfall and repeats the process until the F_j for all targets is less than or equal to zero; see *Line* 6-7.

Algorithm 5: Pseudo-code of TPNP

Input: $x'_{(m,n)}$
Output: $\mathcal{T}_{(m,n)}$
1 $\mathcal{T}_{(m,n)} = \lfloor x'_{(m,n)} \rfloor$
2 **while** *True* **do**
3 **for** *all* $z_j \in Z$ **do**
4 $F_j = E^c - \sum_{(m,n) \in C(z_j)} R_{(m,n)} \times \mathcal{T}_{(m,n)}$
5 **end**
6 **if** $|\{ F_j \mid F_j > 0, \forall z_j \in Z\}| == 0$ **then**
7 Exit
8 **else**
9 $j^* = \arg\max_{j \in Z} F_j$
10 $(m,n)* = \arg\max_{(m,n) \in C(z_{j^*})} R_{(m,n)}$
11 $\mathcal{T}_{(m,n)^*} = \mathcal{T}_{(m,n)^*} + \lceil \frac{F_{j^*}}{R_{(m,n)^*}} \rceil$
12 **end**
13 **end**

A key problem with TPNP is that its solution is still far from optimal. Consider Figure 7.2. The network has three targets: z_1, z_2 and z_3. Each has a shortfall of 200

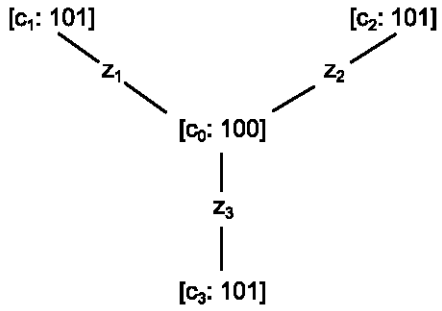


Figure 7.2: An example of TPNP

milliwatts before they have perpetual coverage. There are four cells c_0, c_1, c_2 and c_3. Cell c_0 is able to monitor all three targets whilst cells c_1, c_2 and c_3 are only able to monitor one target. Assume c_0 has an energy recharging rate of 100 milliwatts, c_1, c_2 and c_3 have the same energy recharging rate of 101 milliwatts. Thus, the optimal solution is to place two sensor nodes in cell c_0. However, TPNP will place two sensor nodes in each of the cell c_1, c_2 and c_3. This is because it places sensor nodes in cells that have the maximum recharging rate. Therefore, the optimality gap, explained later, in this scenario is four.

7.3.3 Energy Efficient Node Placement (EENP)

Given the limitation of TPNP, EENP is proposed. The key aim is to place sensor nodes in a manner that reduces the energy shortfall of targets *evenly*. Let $D^j_{(m,n)}$ to indicate the reduced shortfall of target z_j when a sensor node is placed in cell $c_{(m,n)}$. That is,

$$D^j_{(m,n)} = \begin{cases} R_{(m,n)} & \text{if } F_j \geq R_{(m,n)} \\ F_j & \text{if } F_j < R_{(m,n)} \end{cases} \tag{7.5}$$

For each cell $c_{(m,n)}$, EENP calculates the reduced shortfall $D^j_{(m,n)}$ for each target it covers, i.e., $z_j \in Z(c_{(m,n)})$, when placing a sensor node in this cell. Let $D_{(m,n)}$ to indicate the total reduction in shortfall for all targets when a sensor node is placed in cell $c_{(m,n)}$; i.e., $D_{(m,n)} = \sum_{z_j \in Z(c_{(m,n)})} D^j_{(m,n)}$. Let $Z^*(c_{(m,n)}) = \{z_j \mid F_j <$

$R_{(m,n)}, \forall z_j \in Z_k\}$ be a set of targets in Z whereby their shortfall F_j is less than the recharging rate of cell $c_{(m,n)}$. Therefore, the total reduction in shortfall is calculated as follows:

$$D_{(m,n)} = R_{(m,n)}|Z(c_{(m,n)})| - R_{(m,n)}|Z^*(c_{(m,n)})| + \sum_{z_j \in Z^*(c_{(m,n)})} F_j \tag{7.6}$$

Note that $Z(c_{(m,n)})$ returns all the targets covered by cell $c_{(m,n)}$. The second term ensures the targets with $F_j < R_{(m,n)}$ are removed before their correct shortfall is added to $D_{(m,n)}$.

Algorithm 6: Pseudo-code of EENP

Input: $x'_{(m,n)}$
Output: $\mathcal{E}_{(m,n)}$

for *each* $c_{(m,n)} \in C$ **do**
  if $x'_{(m,n)} \neq 0$ **then**
    Add $c_{(m,n)}$ into C'
  end
end
$E_{(m,n)} = \lfloor x'_{(m,n)} \rfloor$
while *True* **do**
  for *all* $z_j \in Z$ **do**
    $F_j = E^c - \sum_{(m,n) \in C(z_j)} R_{(m,n)} \times \mathcal{E}_{(m,n)}$
  end
  if $|\{ F_j \mid F_j > 0, \forall z_j \in Z\}| == 0$ **then**
    Exit
  else
    $D_{MAX} = 0, \quad c_{(m,n)^*} = \emptyset$
    for *each* $c_{(m,n)} \in C'$ **do**
      Calculate $D_{(m,n)}$
      if $D_{(m,n)} > D_{MAX}$ **then**
        $D_{MAX} = D_{(m,n)}$
        $c_{(m,n)^*} = c_{(m,n)}$
      end
      $\mathcal{E}_{(m,n)^*} = \mathcal{E}_{(m,n)^*} + 1$
    end
  end
end

Algorithm 6 presents the steps carried out by EENP. Let $\mathcal{E}_{(m,n)}$ to indicate the number of sensor nodes, as calculated by EENP, that are to be placed in cell $c_{(m,n)}$. EENP first builds a set C' containing cells that has fractional $x'_{(m,n)}$ values; see *Line* 1-5. The cells in C' are also called as *candidate cells*. EENP then rounds down $x'_{(m,n)}$ to $\lfloor x'_{(m,n)} \rfloor$ and calculates the shortfall F_j of all targets; i.e., *Line* 6 and 8-10. After that, EENP calculates $D_{(m,n)}$ for all candidate cells in C'. It then places one sensor node into the candidate cell with the maximum $D_{(m,n)}$; i.e., one that reduces the maximum overall shortfall; see *Line* 15-22. All targets update their shortfall after this sensor node is placed. This process is then repeated until the F_j of all targets is less than or equal to zero; see *Line* 11-12.

7.4 Analysis

This section shows the analysis of GRNP, TPNP and EENP which prove their deterministic worst case bound on the total number of sensor nodes. Let OPT be the objective value of the ILP. Then an algorithm for the MEHNP-PC problem is called a ρ-approximation algorithm, for some $\rho \geq 1$, if the algorithm produces for any solution whose value is at most $\rho \times OPT$. Define the *optimality gap* of a node placement solution to be the objective value of the ILP, i.e., OPT, minus the number of sensor nodes required by the said placement solution. Apart from that, this section also presents the proofs on the time complexity of the proposed algorithms.

Proposition 6. *GRNP is a $(L+1)$-approximation algorithm to the MEHNP-PC problem.*

Proof. Let C' be a set of cells containing fractional number of sensor nodes. Then the number of sensor nodes required to perpetually monitor all targets as calculated by GRNP is

$$\sum_{(m,n)\in C} \mathcal{G}_{(m,n)} = \sum_{(m,n)\in C} \lfloor x'_{(m,n)} \rfloor + |C'| \tag{7.7}$$

where $\lfloor x_{(m,n)} \rfloor$ is the largest integer less than or equal to $x_{(m,n)}$.

Recall that OPT is the optimal objective value returned by ILP. Then the approximation ratio of GRNP is

$$\frac{\sum\limits_{(m,n)\in C} \mathcal{G}_{(m,n)}}{OPT} = \frac{\sum\limits_{(m,n)\in C} \lfloor x'_{(m,n)} \rfloor + |C'|}{OPT} \tag{7.8}$$

$$= \frac{\sum\limits_{(m,n)\in C} \lfloor x'_{(m,n)} \rfloor}{OPT} + \frac{|C'|}{OPT} \tag{7.9}$$

In the above formulation, the expression $\sum_{(m,n)\in C} \lfloor x'_{(m,n)} \rfloor$ is the total number of sensor nodes after round down and takes on a value that is at most OPT. On the other hand, the total number of sensor nodes in the network is at least one. The maximum number of element in C' is no more than $|C|$. Then the approximation ratio of GRNP is

$$\frac{\sum\limits_{(m,n)\in C} \mathcal{G}_{(m,n)}}{OPT} \leq 1 + L \tag{7.10}$$

□

Proposition 7. *TPNP is a $(|Z| + 1)$-approximation solution to the MEHNP-PC problem*

Proof. Let $c_{(m,n)^*}$ be a cell in $C(z_j)$ that can be used to monitor a target z_j. This cell has the maximum recharging rate R_{max} for all cells covering z_j. Let R_{OPT} be the recharging rate of the optimal cell, as calculated by ILP, to monitor target z_j. Lastly, let F_j be the shortfall of target z_j, where $F_j \leq E^c$. Then the approximation

ratio of TPNP is as follows.

$$\frac{\sum_{(m,n)\in C} \mathcal{T}_{(m,n)}}{OPT} \leq \frac{\sum_{(m,n)\in C} \lfloor x'_{(m,n)} \rfloor + \sum_{z_j \in Z} \frac{E^c}{R_{max}}}{OPT} \tag{7.11}$$

$$= \frac{\sum_{(m,n)\in C} \lfloor x'_{(m,n)} \rfloor}{OPT} + \frac{\sum_{z_j \in Z} \frac{E^c}{R_{max}}}{OPT} \tag{7.12}$$

$$\leq 1 + \sum_{z_j \in Z} \frac{\frac{E^c}{R_{max}}}{\frac{E^c}{R_{OPT}}} \tag{7.13}$$

$$= 1 + \sum_{z_j \in Z} \frac{R_{OPT}}{R_{max}} \tag{7.14}$$

$$\leq 1 + |Z| \tag{7.15}$$

□

Proposition 8. *EENP is a* $(\frac{1}{2}|Z| + \frac{3}{2})$*-approximation solution to the MEHNP-PC problem*

Proof. First, two key bounds used in the derivation of EENP's approximation ratio are proved. Assume for the moment that the optimal solution requires only one cell c_{OPT} to cover all targets in Z and its energy harvesting rate is R_{OPT}. Let F_j be the shortfall of target z_j, which has a maximum value E^c. Now consider the set of cells returned by EENP: $C^\alpha = \{c_1, \ldots c_K\}$, where the energy harvesting rate of cell c_k is R_k and it covers targets $Z_k \subseteq Z$. In the best case, C^α has one cell; i.e., c_{OPT}. Otherwise, EENP must have selected another cell c_k that reduces more shortfall than c_{OPT}. Recall, D_k indicates the overall shortfall reduction when placing one sensor node in cell c_k, see Equation (7.5). Thus, we have

$$D_k \geq D_{OPT}, \quad \forall k = 1, \ldots, K \tag{7.16}$$

Note that the targets covered by c_k is a subset of Z; i.e., $Z_k \subset Z$. Therefore, the recharging rate of c_k must be higher than R_{OPT} to ensure Equation (7.16) is satisfied.

We, therefore, have

$$Z_k \subset Z, \quad R_k > R_{OPT}, \quad \forall k = 1, \ldots, K \tag{7.17}$$

Now revise Equation (7.16) to be a function of R_k and R_{OPT} by using Equation (7.6). Let $Z_k^* = \{z_j \mid F_j < R_k, \forall z_j \in Z_k\}$; this set contains targets in Z_k that have a shortfall F_j that is less than the recharging rate of cell c_k. Let $Z^* = \{z_j \mid F_j < R_{OPT}, \forall z_j \in Z\}$ be a set of targets in Z that has a shortfall F_j less than R_{OPT}. Thus, Equation (7.16) can be revised to

$$\begin{aligned} D_k - D_{OPT} =& (R_k|Z_k| - R_{OPT}|Z|) - (R_k|Z_k^*| + \\ & R_{OPT}|Z^*|) + (\sum_{z_j \in Z_k^*} F_j - \sum_{z_j \in Z^*} F_j) \\ & \geq 0 \end{aligned} \tag{7.18}$$

Note that the last term of Equation (7.18) is less than or equal to zero; i.e., $\sum_{z_j \in Z_k^*} F_j - \sum_{z_j \in Z^*} F_j \leq 0$. This is because $R_k > R_{OPT}$, see Equation (7.17), and thus, a target in set Z_k^* may not be in the set Z^*. Therefore, the set of targets in Z_k^* is no more than Z^*. This means we have the following bound for a cell c_k to ensure Equation (7.18) is satisfied,

$$R_k|Z_k| \geq R_{OPT}|Z|, \quad \forall k = 1, \ldots, K \tag{7.19}$$

Another key bound is as follows. Let $C^* \subset C^\alpha$ contain the cells $c_{1^*}, c_{2^*}, \ldots, c_{K^*}$ that have a higher D_{k^*} than D_k. Initially, without loss of generality, assume EENP selects cell c_{1^*}, which has the maximum D_{1^*}. In *Line* 17-20 of Algorithm 6, EENP will reduce the shortfall of targets in $Z(c_{1^*})$ until such point that D_{1^*} is not maximal. EENP will then select the next highest D_{k^*} cell, say c_{2^*}. Note, the D_{2^*} of c_{2^*} will be updated after sensor nodes are placed in cell c_{1^*} if it covers the same targets as c_{1^*}. Critically, c_{2^*} will contain at least one new target. We thus have the following

relationship.

$$(Z_{1^*} \cup Z_{2^*} \cup \ldots \cup Z_{K^*}) \neq Z_k, \quad \forall k \in 1, \ldots, K \tag{7.20}$$

Next, we need to bound the expression $\sum_{k=1}^{K} |Z_k|$. This corresponds to the maximum times targets are covered by cells in C^{α}. In the worst case, each c_k covers all the existing targets covered by cells in C^* plus a new target. In this case, Equation (7.20) can be revised to

$$(Z_{1^*} \cup Z_{2^*} \cup ... \cup Z_{K^*}) \subset Z_k, \quad \forall k \in 1, ..., K \tag{7.21}$$

Without loss of generality, assume $|Z_{1*}| < |Z_{2*}| < \ldots < |Z_{K*}| < |Z_k|$. The maximum number of elements in $|Z_k|$ is $|Z|$, whereby the cell c_k covers all targets in $|Z|$ and have the same recharging rate as c_{OPT}, see Equation (7.19). Hence, in the worst case it has $1 < 2 < 3 < \ldots < |Z|$. Therefore,

$$\sum_{k=1}^{K} |Z_k| \leq 1 + 2 + ... + |Z| \tag{7.22}$$

$$= \frac{|Z|(|Z|+1)}{2} \tag{7.23}$$

As the optimal solution is assumed to consist of one cell, then the optimal number of sensor nodes calculated by ILP will be $OPT = \frac{F^*}{R_{OPT}}$, where F^* is the maximum shortfall for all targets in Z and $F^* \leq E^c$. On the other hand, let F_k^*, where $F_k^* \leq F^*$, be the maximum shortfall for all targets in Z_k. Thus, the number of sensor nodes to be placed in cell c_k is $\frac{F_k^*}{R_k}$. Therefore, using Equation (7.19) in (7.26) and (7.23)

in (7.28), approximation ratio of EENP is as follows:

$$\frac{\sum_{(m,n)\in C} \mathcal{E}_{(m,n)}}{OPT} \leq \frac{\lfloor x'_{(m,n)} \rfloor + \sum_{k=1}^{K} \frac{F_k^*}{R_k}}{OPT} \tag{7.24}$$

$$\leq 1 + \sum_{k=1}^{K} \frac{F_k^*}{R_k} \frac{R_{OPT}}{F^*} \tag{7.25}$$

$$\leq 1 + \sum_{k=1}^{K} \frac{R_{OPT}}{R_k} \tag{7.26}$$

$$\leq 1 + \sum_{k=1}^{K} \frac{R_{OPT}}{\frac{R_{OPT}|Z|}{|Z_k|}} \tag{7.27}$$

$$= 1 + \sum_{k=1}^{K} \frac{|Z_k|}{|Z|} \tag{7.28}$$

$$\leq 1 + \frac{|Z|(|Z|+1)}{2(|Z|)} \tag{7.29}$$

$$= \frac{1}{2}|Z| + \frac{3}{2} \tag{7.30}$$

Therefore, EENP is a $(\frac{1}{2}|Z| + \frac{3}{2})$-approximation algorithm. □

In the proof of EENP's approximation ratio, the optimal solution is assumed to consist of one cell. This assumption is relaxed in the sequel where the optimal solution is a set of cells. Let $C^{OPT} = \{c_1, \ldots, c_Q\}$ be the optimal set of cells returned by ILP. Let the energy harvesting rate of cell c_q be R_q. Note, initially, when no sensor nodes are present, all targets have a shortfall of E^c. Thus, the total number of sensor nodes to be placed in the cells in C^{OPT} is $\sum_{q=1}^{Q} \frac{E^c}{R_q}$, which is less than $\sum_{k=1}^{K} \frac{E^c}{R_k}$. By substituting $\sum_{q=1}^{Q} \frac{E^c}{R_q} \leq \sum_{k=1}^{K} \frac{E^c}{R_k}$ and $F_k^* = E^c$ into Equation (24), it can be shown that the number of sensor nodes to be placed in the cells in C^{OPT} is smaller than the bound $\frac{1}{2}|Z| + \frac{3}{2}$.

The rest of this section presents the running time complexity of the proposed solutions. Recall that they first run the relaxed ILP, i.e., LP^*, to calculate the optimal fractional solution. This ILP contains L decision variables, $|Z|$ constraints, and γ variables in each of its constraints. Here, γ is the number of cells that a sensor

node covers. The solutions then round the fractional solution as per GRNP, TPNP or EENP. Recall that E^c is the energy consumption rate required by targets to be covered perpetually. Let R_{min} be the minimal recharging rate of the cells in the sensing field. Let C' be a set of cells containing fractional number of sensor nodes derived from LP^*. The proposed algorithms have the following time complexity.

Proposition 9. *GRNP has a time complexity of* $\mathcal{O}(L)$.

Proof. GRNP will round up all elements in C' to an integer. As shown in Proposition 6, the number of elements in C' is at most L, which is the total number of cells in the sensing field. Therefore, its running time complexity is $\mathcal{O}(L)$. □

Proposition 10. *TPNP has a time complexity of* $\mathcal{O}(L + \gamma|Z| + 2|Z|^2)$.

Proof. Rounding down all fractional values from LP^* incurs a time complexity of $\mathcal{O}(L)$. Determining the target with the maximum shortfall F_j and iterating over all cells covering it requires $\mathcal{O}(\gamma+|Z|)$ time; see Algorithm 5 *Line* 9-11. Then, it spends $\mathcal{O}(|Z|)$ time to re-calculate F_j for all targets according to the newly placed sensor nodes; see *Line* 3-5. Therefore, the time complexity for rounding up a fractional value to an integer is $\mathcal{O}(\gamma + 2|Z|)$ for each target; see *Line* 3-12. Therefore, TPNP has a time complexity of $\mathcal{O}(L + (\gamma + 2|Z|)|Z|)$. □

Proposition 11. *EENP has a time complexity of* $\mathcal{O}(L + \lceil \frac{E^c}{R_{min}} \rceil |Z|(|Z| + L))$.

Proof. The rounding down step has time complexity $\mathcal{O}(L)$. Next, EENP calculates $D_{(m,n)}$ for each element in C' before placing one sensor node into the sensing field; i.e., Algorithm 6 *Line* 1-5. It then updates the shortfall F_j for all targets after placing a sensor node. Thus, the time complexity of EENP to place one sensor node is $\mathcal{O}(|Z|+L)$; see *Line* 8-23. In the worst case where all cells have the same recharging rate of R_{min}, the maximal number of sensor nodes to be placed is $\lceil \frac{E^c}{R_{min}} \rceil |Z|$. This is because the maximal number of sensor nodes required to cover a target is $\lceil \frac{E^c}{R_{min}} \rceil$. Thus, the running time complexity of EENP is $\mathcal{O}(L + \lceil \frac{E^c}{R_{min}} \rceil |Z|(|Z| + L))$. □

7.5 Research Methodology

To evaluate the performance of all approaches, this chapter shows the experiments conducted in Matlab 2014a on an Intel Core i7 CPU @ 3.5GHz with 8G RAM computer. The parameters of sensor node are based on the datasheet of WaspMote [72]: a sensor node consumes 60mW when active and 0.2 mW when sleeping. The experiments assume all sensor nodes are equipped with an Enocean ECS310 solar cell [73] which has a conversion rate and recharging efficiency of 10% and 50% respectively. This is conservative as compared to other technologies [74]. In addition, the experiments use the average illumination data from Southwest Solar Research Park, Phoenix, Arizona, USA [75]. The parameter values used in the experiments can be found in Table 7.1.

Parameters	**Value**
Battery size	1100 mA [72]
Consumption rate	60mW [72]
Voltage	4V [72]
Solar panel conversion rate	10% [74]
Recharging efficiency	50% [74]
Average recharge rate	0.96 Joules/hour [73][75]

Table 7.1: Simulation Parameters

In the experiments, sensing field is divided into square grid cells. Each cell has an i.i.d recharging rate generated from a random Gaussian distribution with a mean recharging rate of 0.96 Joules per hour, see Table 7.1. All targets are stationary and randomly placed in these cells. Each cell can only have at most one target placed in it. Assumed that all sensor nodes located in the same cell have the same recharging rate $R_{(m,n)}$. Moreover, all sensor nodes have the same sensing radius. It is noticed that transmission cost can be considered by having a higher E^c value.

The experiments investigate the impact of the following parameters on the number of sensor nodes required to cover all targets: sensing radius, target density and network scale. Here, define the target density to be the ratio between the number

of cells with a target over the total number of cells in the sensing field. The three experiments are conducted with one change to the network configuration while others are fixed. Specifically, the network size is fixed to 30×30 when studying the impact of sensing radius and target density. Then both the sensing radius and target density are fixed to observe the impact of network size which is increasing from 30×30 to 300×300. Other configurations will be made specific later. Moreover, the experiments present the optimal result derived from Matlab's ILP solver only in experiments on sensing radius and target density. This is because the ILP is intractable for large networks, e.g., 50×50 cells with 20 targets. Each experiment collects the total number of sensor nodes required to perpetually cover all targets within a sensing field. The results are an average of 200 runs, each with a different topology.

To compare the proposed approximation algorithms, there are two heuristics are derived based on prior works. The first is called Maximum Coverage Node Placement (MCNP). In [63], also see Chapter 2, the authors proposed a heuristic that ensures an event located at a target is detected with a given probability. They defined the 'coverage level' of a cell as the probability that an event will be detected if a sensor node is placed in the cell. They also defined the 'miss probability' of a target to be its required event detection probability minus the 'coverage level' provided by sensor nodes covering it. In the case of energy harvesting WSNs, their solution can be implemented to solve the MEHNP-PC problem by making the following change in technical terms/concepts. Use the 'recharging rate' of a cell instead of 'coverage level'. Replace the 'shortfall' of a target, see Equation 7.4, with 'miss probability'. In the experiments, the resulting MCNP algorithm iteratively places a sensor node in a cell that reduces the maximum overall shortfall $\sum_{j \in Z} F_j$ until all F_j are zero. Its running time complexity can be shown to be $\mathcal{O}(L + \lceil \frac{E^c}{R_{min}} \rceil |Z|(|Z| + L))$, which is the same as EENP. However, in the experiments, the running time of MCNP is significantly higher than EENP in large scale networks. This is because MCNP needs to discover every cell to place one sensor node whilst EENP only considers

the cells or fractional decision variables in the relaxed LP.

The second heuristic is called Efficient Coverage Node Placement (ECNP). This method first adopts the greedy heuristic proposed in [81] to construct a Maximum Weighted Set Cover (MWSC) where the universe is all the targets. Here, the weight of a cell is equivalent to its recharging rate. This greedy heuristic constructs the MWSC by selecting the cell with the maximum weight and also covers the most targets. Then, ECNP places sensor nodes in the cell corresponding to the set cover until the F_j for all target j is zero. It uses the same rule as MCNP. That is, it selects the sensor node that reduces the maximum overall shortfall in each iteration.

7.6 Results

This section first presents the results concerning sensing radius and target density for a network with 30×30 cells. It then shows the impact of different network scales.

7.6.1 Sensing Radius

In this experiment, there are nine targets are randomly placed in the sensing field. The resulting sensing area thus has a target density of 1%. After generating a random topology, the sensing radius is increasing from three to 30 with an interval of three. In order to clearly indicate the different results for each approach, the ratio of sensor node numbers versus the ILP solution is also plotted.

From Figure 7.3, it can be seen that the number of sensor nodes decreases from around 23 to 3 with increasing sensing radius. The reason is because a large sensing radius means fewer sensor nodes are required to cover all targets. It also shows that the number of sensor nodes as determined by ILP is a constant when the sensing radius is greater than 18 cells. This is because a cell with the maximum recharging rate is able to cover all targets. Thus, it simply places multiple sensor nodes in that cell.

Figure 7.4 shows a comparison between EENP, TPNP, GRNP, MCNP and ECNP

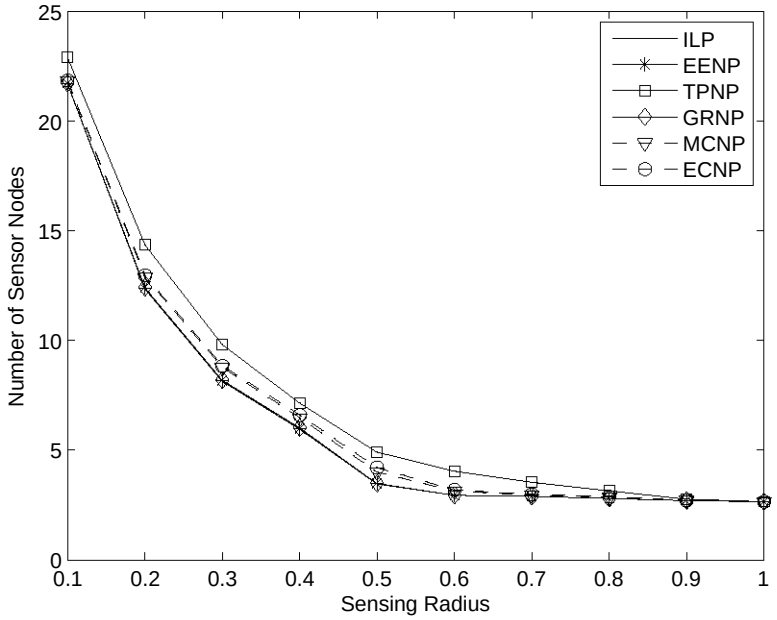


Figure 7.3: Impact of sensing radius

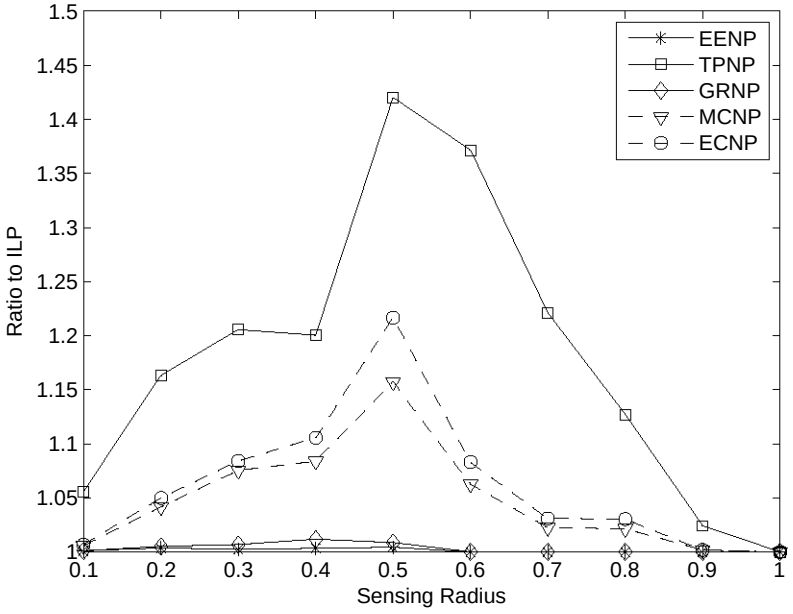


Figure 7.4: Impact of sensing radius ratio to ILP

to ILP. The results of EENP and GRNP are close to the ILP with a gap of 1% and 2%, respectively. On the other hand, the gap for TPNP can be as high as 43%. This is because it greedily fulfills one target without considering others. For MCNP and ECNP, their gap is 14% and 22%, respectively. This is because MCNP and ECNP greedily select cells that reduce the maximum shortfall, which may miss the optimal locations. For example, consider three targets A, B and C that require 100 milliwatts to achieve perpetual coverage. There are two cells x and y with a recharging rate of 55 and 90 milliwatts respectively. Cell x is able to cover all targets whilst y only covers targets A and B. The optimal placement to cover all targets is, obviously, to place two sensor nodes into cell x. However, MCNP and ECNP will place one sensor node in cell y and two sensor nodes in cell x. This results in an optimality gap of one sensor node. The reason is that the weight of cell y is 180, which greater than cell x with a weight of 150. Therefore, they choose cell y instead of x. Moreover, ECNP may have discarded potentially optimal cells when calculating the minimal set cover, which leads to a further 12% gap as compared to MCNP.

From Figure 7.4, it also shows that the optimality gap of TPNP, MCNP and ECNP increases when the sensing radius ranges from 3 to 15 cells and peaks when the sensing radius is 15 cells. This is because the number of overlapped cells, e.g., a cell covering multiple targets, and their overlapped level, e.g., the number of targets a cell can cover, increases. Now reconsider the above example where two cells x and y are covering three targets A, B and C. All targets require 100 milliwatts to achieve perpetual coverage whilst cell x and y has a recharging rate of 55 and 90 milliwatts respectively. Specifically, the target C is covered by cell x whilst targets A and B are covered by cell y. In this case, the number of sensor nodes calculated by TPNP, MCNP and ECNP is the same as the optimal result. However, when increasing the sensing range of x to cover all targets; i.e., A, B and C, the required number of sensor nodes is increased by one. This is because the weight of cell y, which is a suboptimal cell, is higher than x. On the other hand, the gap is reduced

when increasing the sensing radius from 15 to 30 cells. This is because there are more cells with a high recharging rate that are able to cover all targets. Thus, it increases the probability that TPNP, MCNP and ECNP will select these cells.

7.6.2 Target Density

In this experiment, the sensing radius is fixed to 9 cells. The metric recorded in this experiment is the number of sensor nodes required to ensure perpetual coverage when increasing the number of targets from nine to 180 targets with an interval of nine. In other words, the target density is increased from 1% to 20% with an interval of 1%.

Figure 7.5 shows the number of sensor nodes required by ILP, EENP, TPNP, GRNP, MCNP and ECNP with increasing number of targets. It can be observed that the number of sensor nodes required by ILP, EENP and GRNP plateaus at 12 sensor nodes when dispersing 27 or more targets. This is because 12 sensor nodes are sufficient to cover the entire sensing field perpetually. Hence, due to omni-directional sensing, targets can be placed anywhere without requiring any additional sensor nodes. On the other hand, TPNP and MCNP need 17 sensor nodes to cover the sensing field whilst MCNP requires 19 sensor nodes. This is because MCNP is a greedy algorithm and may miss the optimal cells.

7.6.3 Network Scale

In this experiment, both sensing radius and target density are fixed to 25 cells and 0.5% respectively. The network dimension l is increasing from 30 to 300 cells with an interval of 30 cells. Note, the number of targets increases when scaling the network size. This experiment does not present results from the ILP because the network sizes under consideration are computationally intractable.

From Figure 7.6, it can be seen that there are more sensor nodes with increasing network scale. This is because there are more targets, which requires more sensor

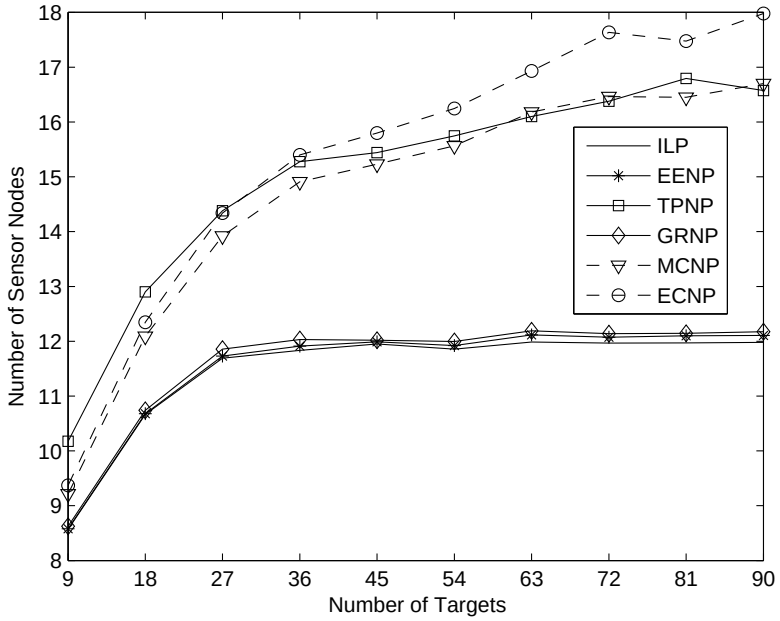


Figure 7.5: Number of sensor nodes versus number of targets

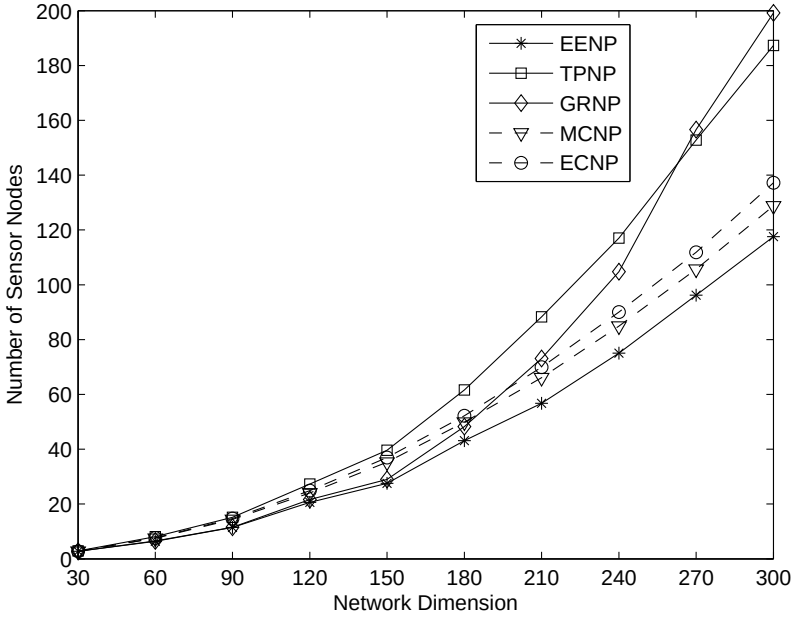


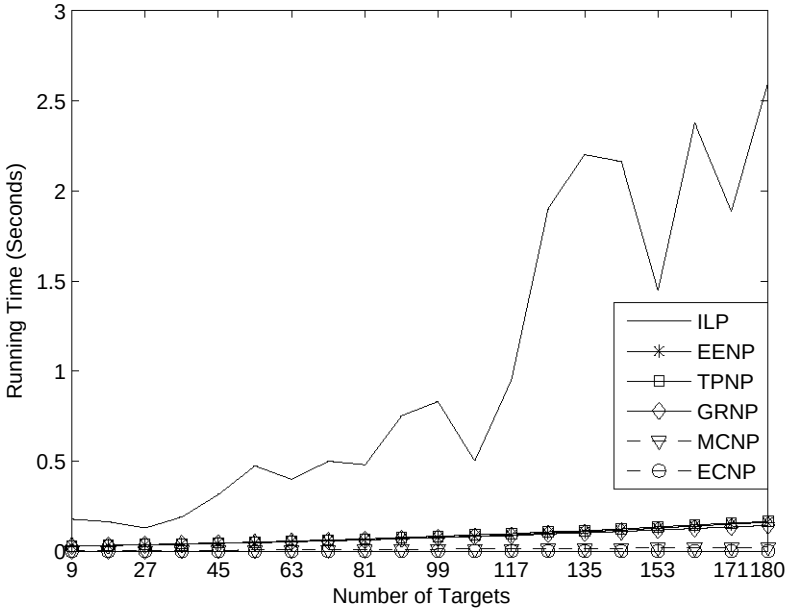
Figure 7.6: Number of sensor nodes versus network dimension ($l \times l$)

nodes to cover them. It also shows a similar trend between ECNP, MCNP and EENP. Therefore, the number of additional sensor nodes required for ECNP, MCNP and EENP are similar when increasing the network size. However, for GRNP, the number of sensor nodes exceeds other algorithms and reaches 200 when the network size is 300×300 cells. This is because it greedily rounds up all fractional LP results. This result also verifies the poor approximation ratio of GRNP. Therefore, GRNP is not suitable for large scale networks, especially when the sensing range is small, e.g., each sensor node covers only 5% of the entire sensing field.

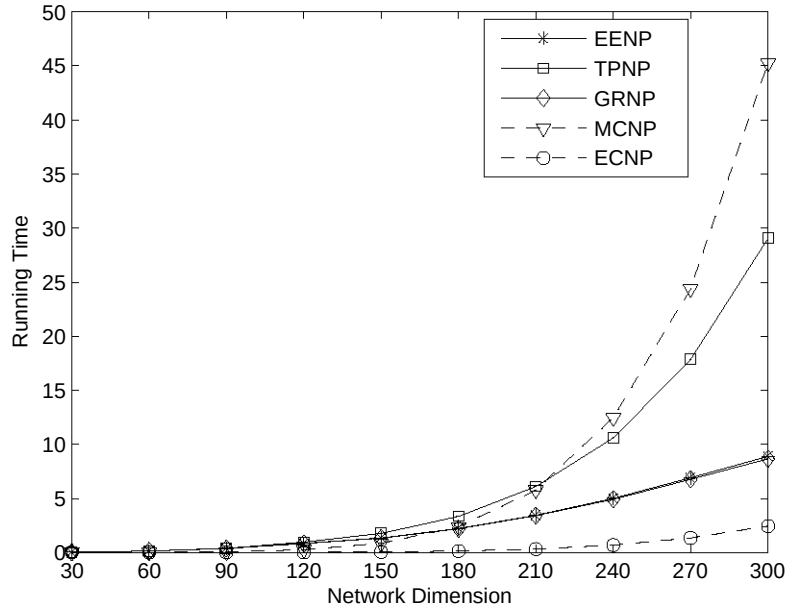
7.6.4 Running Time

This section presents the results on running time of all proposed algorithms. From Figure 7.7a, it can be seen that the running time of EENP, TPNP, GRNP, MCNP and ECNP gradually increases with the number of targets. This is because EENP, TPNP and GRNP first run LP^*; see Section 7.4. The number of constraints in LP^* as well as the iterations incurred by MCNP and ECNP to select cells to place sensor nodes is proportional to the number of targets. On the other hand, as expected, the running time of ILP increases significantly from 0.2 to 2.5 seconds with increasing target density.

Figure 7.7b shows the running time of MCNP increases quadratically with the network size. This is because MCNP compares all cells in the sensing field to determine the location for each sensor node. Thus, MCNP is not suitable for large scale networks. The running time of ECNP is significantly lower than MCNP. The reason is because ECNP is initialized with the maximal weight set cover, see Section 7.5. ECNP then places sensor nodes into cells that correspond to the set cover. As the set cover is minimal by definition, ECNP checks significant fewer number of cells as compared to MCNP. Another observation from Figure 7.7b is that the running time of EENP, TPNP and GRNP increases with increasing network size. This is because there are more decision variables in the relaxed LP, i.e., LP^*. Moreover,



(a)



(b)

Figure 7.7: Running time with increasing (a) target density; (b) network scale.

both Figure 7.7a and 7.7b show the running time of EENP and TPNP are close to GRNP. Thus, the running time incurred by rounding fractional solution to integer is negligible as compared to the running time of LP^*. Figure 7.7b also shows that the running time of TPNP is higher than EENP, which agrees with the analytical results in Section 7.4. Therefore, EENP outperforms TPNP, GRNP, MCNP and ECNP in terms of the required number of sensor nodes. EENP also has a lower computation time as compared to MCNP.

7.7 Conclusion

This chapter has presented the first work that considers the energy harvesting node placement for energy neutral operation problem; i.e., MEHNP-PC. The said problem is formulated as an ILP and three rounding algorithms, called GRNP, TPNP and EENP, are proposed. Critically, these algorithms have a provable approximation ratio. In particular, the required number of sensor nodes by these algorithms are at most $L+1$, $|Z|+1$ and $\frac{1}{2}|Z|+\frac{3}{2}$ times the optimal. Moreover, the experiments results show that EENP is near optimal, with a 1% gap in terms of the number of sensor nodes required to cover all targets.

A key limitation, however, is that these algorithms assume sensor nodes have a direct link to the base station/sink. Thus, to relax this assumption, next chapter considers the MEHNP problem with network connectivity constraint. It ensures active sensor nodes have a multi-hop path to the base station/sink.

Chapter 8

Node Placement and Scheduling for Complete Targets Coverage and Connectivity

The aim of this chapter is to revisit the problem of the previous chapter but with consideration for connectivity. Specifically, this chapter addresses the MEHNP-ENCC problem. In a nutshell, the MEHNP-ENCC problem jointly considers the duty cycling and node placement problems to ensure complete targets coverage, network connectivity and energy neutral operation. This means the objective of the MEHNP-ENCC problem is to determine the locations to place the minimal number of sensing and relaying nodes such that all targets are completely covered and the network is connected and has energy neutral operation. Apart from that, the solutions for the EHNP-ENCC problem also determine the activation schedule of sensor nodes whereby the active sensor nodes maintaining complete targets coverage and network connectivity.

The first approach considered is to formulate the problem as a MILP. The main

decision variables are the set of candidate locations that ensure complete targets coverage and connectivity when a sensor node is placed in them. However, this incurs high computation cost. Therefore, a greedy heuristic is proposed to reduce the number of candidate locations or decision variables considered by the MILP. This solution is called Greedy MILP or GMILP. However, the running time of GMILP is still a concern. To this end, two heuristics, namely, DirectSearch and GreedySearch, are proposed to iteratively determine the locations to place sensor nodes for sensing and relaying. Simulation results show that DirectSearch requires 20% more sensor nodes than MILP whilst this value is 10% for GreedySearch and GMILP. Moreover, the running time of DirectSearch and GreedySearch is an order faster than GMILP and four orders faster than MILP.

8.1 Network Model

In this chapter, the sensing field is modeled as a grid structure as shown in Figure 8.1. Let λ be the length of the interval between any adjacent grid points on the same row or column. Let l be the dimension of the sensing field, and is equal to the total number of grid points in a row or column. The set L is a collection of grid points of size $|L| = l \times l$. The set of grid points containing at most one target is recorded in the set Z; e.g., for Figure 8.1, it has $Z = \{6, 10\}$. Let n and m be the total number of grid points and targets in the sensing field. Let B to denote the sink, e.g., point 4. Note that in this chapter, there is only *one* sink in the sensing field. Both the sink and targets have a fixed location.

Let r and r' be the uniform sensing and communication radius of all sensor nodes, respectively. Assume the communication range is twice the sensing radius, i.e., $r' = 2r$. Let $Z(i)$ be a function that returns the set of targets within the sensing range of grid point i. Conversely, $L(j)$ is a function that returns the set of grid points that are able to cover target j. For example, in Figure 8.1 where the sensing radius is equal to λ, the grid points that cover targets 6 and 10, respectively, are

$L(6) = \{2, 5, 6, 7, 10\}$ and $L(10) = \{6, 9, 10, 11, 14\}$. Let $N(i)$ return the set of grid points that are within the communication range of grid point i. Define $C_k \subseteq L$ to be a subset of grid points that are able to cover all targets, where $k = 1, \ldots, K$. Let $\delta(C_k, i)$ be an indicator function that returns one if grid point i is in the subset C_k, otherwise it returns zero. Let Ω to be a collection of C_k; i.e., $\Omega = \{C_1, \ldots, C_K\}$. Note that the total number of subsets C_k, i.e., the value of K, is equal to two to the power of the number of grid points that see at least one target. In Figure 8.1, possible subsets C_k include $\{2, 9\}$, $\{6\}$ or $\{2, 6, 9, 10\}$. The total number of subsets K or $|\Omega|$ is $2^8 = 256$. Let $d(i, h)$ to be the shortest Euclidean distance between any pair of grid points i and h. For example, the distance from targets 6 and 10 to the sink can be shown to be $d(6, 4) = \sqrt{5}\lambda$ and $d(10, 4) = 2\sqrt{2}\lambda$, respectively.

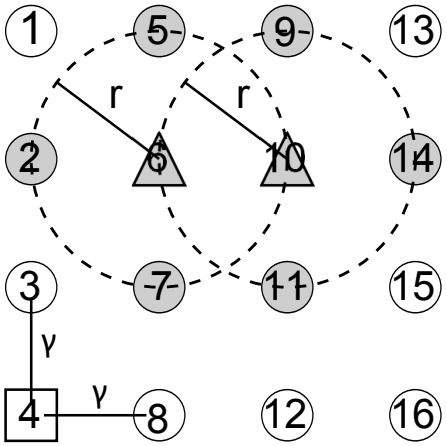


Figure 8.1: An example sensing field. Circles are grid points, triangles are grid points that contain a target, the square denotes the sink. Filled patterns are grid points that are able to cover at least one target.

Assume that time is divided into slots of one unit length. Let c_k be the active time of the sensor nodes in C_k. Thus, the *total* activation time for sensor nodes on the grid point i in each time slot is $x_i = \sum_{k=1}^{K} \delta(C_k, i) c_k$. Let E^m (Watts) be the energy consumption rate when a sensor node is in the active state. The energy consumed when sensor node is in the sleep state is assumed negligible. In addition, assume each grid point has an i.i.d energy harvesting rate R_i (Watts). This is the energy harvesting rate of sensor nodes if they are placed on grid point i. Let G (bit/s) be the data generation rate when a sensor node monitors one target. Let

E^r and E^t (Joule/Bit) to be the consumed energy when a sensor node receives and transmits one bit of information, respectively. The energy consumed by a sensor node to forward one bit of information, i.e., E^f (Joule/Bit), is thus $E^f = E^r + E^t$. Let E_i^c be the energy consumption rate of sensor nodes deployed on grid point i. Assume a grid point can deploy one of two types of sensor nodes: sensing or relaying. The energy consumption rate of sensing nodes is equal to the energy consumption rate for monitoring targets and forwarding data; i.e.,

$$E_i^c = (E^m + E^t \sum_{j \in Z(i)} G)x_i + E^f \sum_{h \in N(i)} f_{hi} \tag{8.1}$$

where f_{hi} is the data forwarding rate from grid point h into i. Note that if a sensor node is only responsible for relaying data, then its energy consumption rate is only represented by the last term of Equation (8.1).

8.2 Problem Statement

This section shows the formal description of the MEHNP-ENCC problem. Consider a sensing field, partitioned into a grid, with known targets and sink location. Each grid point has an i.i.d energy harvesting rate should a sensor node be placed in it. The *aims* of MEHNP-ENCC problem is to determine the minimum number of sensor nodes and their location such that all targets are continuously monitored whilst ensuring energy neutral, complete targets coverage and connectivity. Note that a WSN is saying *energy neutral* when the energy harvesting rate of each sensor node is higher than its energy consumption rate attributed to monitoring and relaying data [25]. Moreover, a WSN is connected if there is at least one path from a sensor node to the sink. Lastly, each grid point can have multiple sensor nodes.

The MLIP formulation for the MEHNP-ENCC problem is shown as follows. Let y_i be the number of sensor nodes to be placed on grid point i. Energy neutral

complete target coverage is achieved when $y_i \geq \lceil \frac{E_i^c}{R_i} \rceil$ for all $i \in L$. We have

$$\text{MIN} \sum_i y_i \tag{8.2}$$

subject to:

$$\sum_{k=1}^{K} c_k \geq 1 \tag{8.3}$$

$$G \sum_{j \in Z(i)} x_i + \sum_{h \in N(i)} f_{hi} = \sum_{h \in N(i)} f_{ih} + f_{iB}, \quad \forall i \in L \tag{8.4}$$

$$\sum_i f_{iB} = G \sum_i \sum_{j \in Z(i)} x_i \tag{8.5}$$

$$E^m x_i + E^t G \sum_{j \in Z(i)} x_i + E^f \sum_{h \in N(i)} f_{hi} - y_i R_i \leq 0, \quad \forall i \in L \tag{8.6}$$

Constraint (8.3) ensures complete target coverage for at least one time unit. Constraints (8.4) and (8.5) ensure flow conservation and they also ensure connectivity from each sensor node to the sink. Note that the data forwarding rate f_{iB} will be zero if cell i is not directly connected to the sink. Constraint (8.6) ensures the energy consumption rate of a grid point is less than the total recharging rate of placed sensor nodes.

The number of decision variables in this MILP is $l^2 + |\Omega| + l^4$, and the number of constraints is $2(1 + l^2)$. A key problem is the exhaustive search for all set covers in Ω. Unfortunately, this is computationally prohibitive [37]. Also, due the size of Ω, the MILP has a high computational cost; e.g., the worst $|\Omega|$ value for the example shown in Figure 8.1 is 2^8. Consequently, the heuristics proposed in the next section are required for large scale WSNs. This section concludes with the following proposition.

Proposition 12. *The MEHNP-ENCC problem is NP-complete.*

Proof. This proof shows that an instance of MEHNP-ENCC is equivalent to the NP-complete, Minimum Set Cover Problem [81]. Let all grid points contain a target,

and the set of targets Z be the ground set. Each grid point is a set that contains targets it covers when a node is placed in it, i.e., $Z(i)$. In addition, all grid points have the same potential recharging rate of one Watt, and E_i^c is one Watt. It thus has an instance of the minimum set cover problem; i.e., determine the minimum number of grid points that cover all targets. This completes the proof. □

8.3 Solutions

There are three solutions for the MEHNP-ENCC problem are proposed in this section. Namely, Greedy MILP (GMILP), DirectSearch and GreedySearch. The GMILP algorithm uses the maximum weighted set cover or MWSC heuristic of [81] to reduce the number of set covers or decision variables considered by the MILP. Both DirectSearch and GreedySearch rely on a straight line that connects targets and the sink. Intuitively, this ensures the shortest path is used to reach the sink. They then assign a weight to each grid point, which is proportional to the grid point's potential recharging rate and the lines, in terms of length, it covers. If a grid point covers multiple lines, then its weight includes the portion covered for each line. The heuristics then determine the grid points to deploy sensor nodes for monitoring targets or to relay data. However, as will be shown in Section 8.4, the sensor nodes required by DirectSearch can be arbitrarily far from the optimal if the recharging rate of a grid point on a line is made arbitrarily small. This is because DirectSearch only considers the grid points around the straight line connecting targets and the sink.

8.3.1 Greedy MILP (GMILP)

Let $\mathcal{U}$ be the universe of elements and a collection of subsets $\mathcal{S} = \{S_i \mid i = 1, \ldots |\mathcal{S}|\}$. Each S_i is a subset of $\mathcal{U}$ and has a weight w_i. In each iteration, MWSC selects a S_i with the maximum $w_i \times |S_i|$ value into a set $\mathcal{P}$, which contains the solution. It then removes the elements in S_i from the universe $\mathcal{U}$. After that it updates all subsets $\mathcal{S}$

and proceeds to the next iteration until $\mathcal{U}$ is empty.

As mentioned earlier, the size of Ω is prohibitively large, which makes solving the MILP challenging. A logical solution is thus to reduce the search space by pre-processesing the input to yield only "critical" set covers. This involves two steps. First, GMILP uses MWSC to determine a group of grid points, called Candidate Points (CPs), that can be used to sense the entire sensing field; see Algorithm 7. Let L be the ground set. Recall that $L(i)$ is the set of grid points that can cover target i, which in this case, every grid point $i \in L$ is considered a target. GMILP then sets the weight of $L(i)$ to be equal to the potential recharging rate R_i. At this point, GMILP has a set cover instance. It thus calls MWSC to yield CP. In each iteration, it selects a grid point with the maximum $w_i \times |L(i)|$ value into the set CP; see *Line* 6. Note that if the communication radius of nodes placed in the grid points of CP is set to twice their sensing radius, then as shown in [27], the nodes form a connected network.

Algorithm 7: Pseudocode to construct CP

Input: L, R_i
Output: CP

```
1 L' = L
2 while L' ≠ ∅ do
3 |   for all i ∈ L do
4 |   |   w_i = R_i
5 |   end
6 |   i* = arg max_{i∈L} w_i |L'(i)|
7 |   CP = CP ∪ {i*}
8 |   L' = L' − L(i*)
9 end
```

The next step is to extract as many disjoint set covers from CP as possible in order to monitor all targets; see Algorithm 8. Let $C'_k \subseteq CP$ be a subset of grid points that cover all the targets, where $k = 1, \ldots, K'$. The collection of C'_k is denoted as Ω'; i.e., $\Omega' = \{C'_1, \ldots, C'_K\}$. Again, GMILP will use MWSC to construct each C'_k. To do this, let the set of targets Z be the ground set. Recall that each $Z(i)$ is a subset that contains the targets within the sensing range of grid point i, where $i \in CP$.

GMILP sets the weight of each subset $Z(i)$ to be the potential recharging rate of grid point i. It then calls MWSC to compute a set cover C'_k; see *Line* 6-13. After that, it removes the grid points in C'_k from CP, and repeat the process until the remaining grid points in CP are not able to ensure complete targets coverage.

Algorithm 8: Pseudocode to constuct Ω'

Input: Z, R_i, CP
Output: Ω'

```
1  k = 0
2  while true do
3  |   k = k + 1
4  |   C'_k = ∅
5  |   Z' = Z
6  |   while Z' ≠ ∅ do
7  |   |   for all i ∈ CP do
8  |   |   |   w_i = R_i
9  |   |   end
10 |   |   i* = arg max_{i∈CP'} w_i|Z'(i)|
11 |   |   C'_k = C'_k ∪ {i*}
12 |   |   Z' = Z' − Z(i*)
13 |   end
14 |   CP = CP − C'_k
15 |   if {Z(j) | ∀j ∈ CP} ≠ Z then
16 |   |   Stop
17 |   end
18 end
19 Ω' = {C'_k | ∀k}
```

At this point, GMILP has successfully pre-processed the input to reduce the search space of the MILP in Section 8.2. Specifically, it has $2|CP| + |CP|^2$ decision variables in the objective function and $2(1 + |CP|^2)$ constraints. This is because the number of integer decision variables is $|CP|$ and the number of connections f_{ih} is $|CP|^2$. On the other hand, there are at most $|CP|$ set covers in the collection Ω' after the second step. Consequently, GMILP significantly increases the computation efficiency of MILP. For example, in Figure 8.1, the number of decision variables in MILP is 1296; i.e., $4^2 + 2^8 + 4^4$. However, if using GMILP, this value is reduced to 80. This is because the number of elements in CP is at most eight in this example. In addition, as will be demonstrated in Section 8.6.3, the running time of MILP

is orders of magnitude higher than GMILP with increasing network dimension and number of targets.

8.3.2 DirectSearch

This subsection presents the first of two heuristics that rely on a straight line from targets to the sink. It starts by presenting the weight assignment procedure. Let P_j be a straight line connecting target j and the sink. Let $d'(i, P_j, r)$ be the portion or length of line P_j that is covered by grid point i given sensing radius r. The weight of a grid point i, namely w'_i, is defined as,

$$w'_i = \sum_{j \in Z} R_i \times d'(i, P_j, r) \tag{8.7}$$

Note that the definition of 'weight' in Equation 8.7 is different from the one used in Section 8.3.1.

In order to calculate $d'(i, P_j, r)$, one needs to consider two cases: (1) full, and (2) partial cover. Referring to Figure 8.2, in the full cover case, line P_j has two intersecting points with the boundary of the sensing field of grid point i; see Figure 8.2a. In the partial cover scenario case, the start or end point of line P_j is in the sensing field of grid point i; see Figure 8.2b. Let D be a virtual point on BC where AD is orthogonal to BC. Then $d'(i, P_j, r)$ for each case can be calculated by using the Pythagoras theorem as follows:

Case 1: $BC^2 \geq AB^2 + AC^2$.

$$d'(A, P_C, r) = 2\sqrt{(r^2 - AD^2)} \tag{8.8}$$

Case 2: $AB^2 \geq BC^2 + AD^2$.

$$d'(A, P_C, r) = \sqrt{r^2 - AD^2} - (BD - BC) \tag{8.9}$$

$$= \sqrt{r^2 - AD^2} - \sqrt{AB^2 - AD^2} + BC \tag{8.10}$$

Note that the length of each side of the triangle ΔABC, i.e., AB, AC and BC, is the distance between grid points A, B and C. The length of AD given the area of ΔABC can be calculated by using Heron's [82] formula as follows.

$$\Delta ABC = \sqrt{p(p - AB)(p - AC)(p - BC)} \tag{8.11}$$

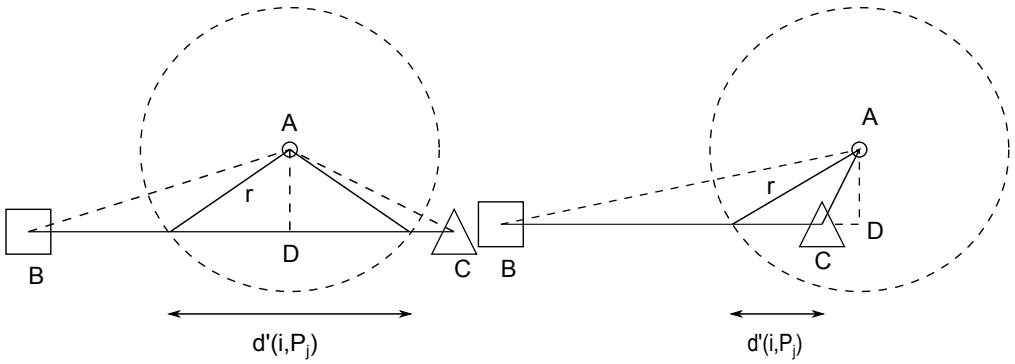
where

$$p = \frac{AB + AC + BC}{2} \tag{8.12}$$

Thus, the length of AD is

$$AD = \frac{\Delta ABC}{BC} \tag{8.13}$$

$$= \frac{\frac{1}{4}\sqrt{4AB^2AC^2 - (AB^2 + AC^2 - BC^2)^2}}{BC} \tag{8.14}$$



(a) (b)

Figure 8.2: An example of (a) full cover, and (b) partial cover. A is a grid point, B is the sink and C is the target. D is a virtual point where AD is orthogonal to BC.

Figure 8.3 shows an example of assigning a weight to grid points 1, 6, and 7. Grid points 10 and 15 are targets whilst 4 is the sink. The sensing radius is equal to the distance between each grid point, i.e., λ. All grid points have a uniform recharging rate of 100 milliwatt. There are two lines P_{10} and P_{15}, each connecting a target and the sink. In this scenario, the weight of grid point 1, i.e., w'_1, is zero because it does not cover any line. On the other hand, the weight of grid points 6 and 7 is 0.949λ

and 2.919λ, respectively. This is because point 7 covers the line from target 10 and 15 to the sink.

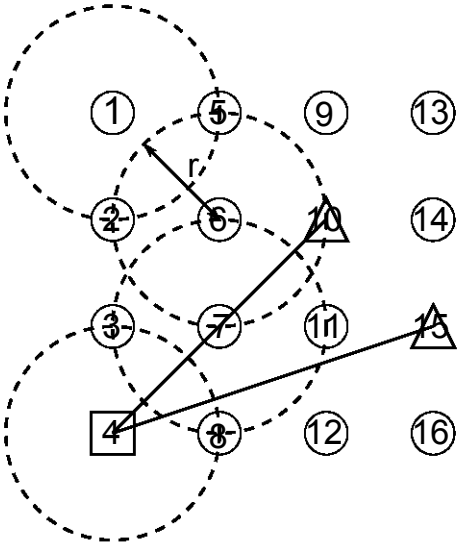


Figure 8.3: The weight of grid points is shown next to each circle; all grid points have a uniform recharging rate of 100 milliwatt; $\Delta = r$

Algorithm 9 presents the details of DirectSearch. It first calculates the weight of all grid points as per Equation (8.7); see *Line* 1-3. It then determines the grid points to deploy sensing or relaying nodes; see *Line* 4-16. Let CL be a set containing the grid points to deploy sensing nodes whilst RL contains the grid points to deploy relaying nodes. Let i^* be the grid point with the minimum weight in L; see *Line* 5. Recall that L is the collection of grid points in the sensing field. DirectSearch will simply remove i^* from the set L if all lines within i^*'s sensing range can be covered by other grid points in L or already determined CL and RL. To do this, it divides each line P_j into p 'segments'. Let $Q(i, P_j)$ be a function that returns these segments for line P_j that are within the sensing range of i. Therefore, $\{Q(i^*, P_j) \mid \forall j \in Z\}$ is the set of segments covered by i^*. If any segment in $\{Q(i^*, P_j) \mid \forall j \in Z\}$ is covered only by i^*, then i will be added into the set RL or CL. Specifically, if all targets in $Z(i^*)$ can be covered by other grid points, then i^* will be added into the set RL; see *Line* 9. Otherwise, if any target in $Z(i^*)$ is covered only by grid point i^*, DirectSearch will add i^* into the set CL. DirectSearch then removes i^* from the set L and repeats the previous steps *Line* 5-15 for the next grid point in L' with

the minimal weight until L is empty. Lastly, DirectSearch determines the energy consumption rate, i.e., E_i^c, for sensing and relaying nodes as per Equation (8.1). The number of sensor nodes required in each grid point is equal to $\lceil \frac{E_i^c}{R_i} \rceil$.

Algorithm 9: Pseudocode for DirectSearch

Input: L, R_i, Z, r
Output: CL, RL

for $\forall i \in L$ **do**
  $w_i' = \sum_{j \in Z} R_i \times d'(i, P_j, r)$
end
while $L \neq \emptyset$ **do**
  $i^* = \arg\min_{i \in L} w_i'$
  if $\{Q(i^*, P_j) \mid \forall j \in Z\} \subseteq \{Q(i, P_j) \mid \forall j \in Z, \forall i \in L \cup CL \cup RL, i \neq i^*\}$ **then**
    Continue
  else
    if $Z(i^*) \subseteq \{Z(j) \mid \forall j \in L \cup CL, j \neq i^*\}$ **then**
      $RL = RL \cup \{i^*\}$
    else
      $CL = CL \cup \{i^*\}$
    end
  end
  $L = L - \{i^*\}$
end

8.3.3 GreedySearch

A key drawback of DirectSearch is that it only considers the grid points around the straight line connecting targets and the sink. Therefore, a lot of unnecessary sensor nodes will be deployed if all these grid points have a low potential recharging rate; see Proposition 15 in Section 8.4. To overcome this drawback, GreedySearch is proposed, which considers more grid points as compared to DirectSearch to place relay nodes.

GreedySearch has three phases. In the first phase, it determines a set of grid points CL to place sensing node in order to monitor targets. In the second phase, it determines sets of grid points $\{RL_k\}$, where $k = 0, \ldots, K$, to place relaying nodes. These nodes forward data from those in CL to the sink. In the last phase,

GreedySearch calculates the energy consumption rate E_i^c for each grid point i, and computes the required number of sensor nodes as $\lceil \frac{E_i^c}{R_i} \rceil$.

The pseudocode of the first phase is shown in Algorithm 10. It first calculates the weight of all grid points based on the line that connects each target and the sink; see *Line* 1-3. Let i^* be the grid point that has the minimum weight in L. GreedySearch will simply remove i^* from L if all targets within the sensing range of i^* can be covered by other grid points; see *Line* 11. Recall that $L(i)$ returns the set of grid points within the sensing range of i. Otherwise, grid point i^* is added into the set CL; see *Line* 6-10. GreedySearch then repeats *Line* 5-11 for the next grid point in L with the minimal weight until L is empty.

Algorithm 10: Pseudocode for GreedySearch Phase 1

Input: L, R_i, Z, r
Output: CL
1 **for** $\forall i \in L$ **do**
2 $\quad w_i' = \sum_{j \in Z} R_i \times d'(i, P_j, r)$
3 **end**
4 **while** $L \neq \emptyset$ **do**
5 $\quad i^* = \arg\min_{i \in L} w_i'$
6 $\quad$ **if** $L(i^*) \cap Z \subseteq \{L(i) \mid \forall i \in L \cup CL, i \neq i^*\}$ **then**
7 $\quad\quad$ Continue
8 $\quad$ **else**
9 $\quad\quad CL = CL \cup \{i^*\}$
10 $\quad$ **end**
11 $\quad L = L - \{i^*\}$
12 **end**

In the second phase, see Algorithm 11, GreedySearch iteratively determines the set of grid points in RL_k to connect grid points in the set RL_{k-1}. Initially, $k = 0$ and RL_k equals CL; i.e., $RL_0 = CL$. In order to connect the nodes in RL_k and RL_{k-1}, GreedySearch calls Algorithm 10 in each iteration. Specifically, Algorithm 10 is provided with the following inputs: the set of grid points L, potential recharging rate R_i, the set of relay points in RL_{k-1} and the communication radius r'; note, it uses the communication range because the aim is to 'connect' instead of to 'cover'.

In each iteration k, GreedySearch looks for a set of grid points within the com-

munication range of nodes in RL_{k-1} that have a high weight. The weight of these grid points is calculated based on the line that connects each node in RL_{k-1} and the sink. Consequently, the search space of GreedySearch is bigger than that of DirectSearch. GreedySearch stops if a set RL_k only contains the sink B; i.e., *Line* 6. This means all relay points in the sensing field have a path to forward data to the sink. Otherwise, it repeats *Line* 4-10 to determine the next set of relay points.

Algorithm 11: Pseudocode for GreedySearch Phase 2

Input: L, R_i, CL, r'
Output: $\{RL_k\}$

```
1  k = 0
2  RL_k = CL
3  while true do
4  |   k = k + 1
5  |   RL_k = Algorithm-10(L, R_i, RL_{k-1}, r')
6  |   if RL_k == {B} then
7  |   |   Stop
8  |   else
9  |   |   Continue
10 |   end
11 end
```

In the last phase, see Algorithm 12, GreedySearch calculates the energy consumption rate for all grid points. To do this, let F_i (bit/s) be the data forwarding rate at grid point i. This rate is dependent on the number of sensing nodes it is relaying for. Note, initially, only sensor nodes in the set CL have a non-zero data forwarding rate that corresponds to the number of targets they cover; see *Line* 1-4. It then calculates the energy consumption rate of relay points in order; i.e., RL_0 to RL_K. Let i^* be the grid point in the set RL'_k with the maximum potential recharging rate; see *Line* 8. The grid point i^* is responsible relaying data for all its neighbours in RL'_{k-1}. This is because a grid point in RL_{k-1} will select one of its neighbours in RL_k with the highest R_i to forward its data. The data forwarding rate at i^* is increased by F_h if i^* is relaying data for grid point h. In other words, if i^* is forwarding data for multiple grid points in the set RL_{k-1}, then the data forwarding rate at i^* is increased by $\sum_{h \in \{RL_{k-1} \cup N(i^*)\}} F_h$, where $N(i^*)$ are neighbours of i^*; see *Line* 9.

Then the energy consumption rate $E^c_{i^*}$ of sensor nodes on grid point i^* is increased by $\sum_{h\in\{RL_{k-1}\cup N(i^*)\}} F_h E_t$. A grid point will be removed from the set RL_{k-1} after GreedySearch assigns a relay point for it; see *Line* 11. This is to avoid redundancy during transmission. GreedySearch then removes i^* from the set RL'_k and repeats *Line* 8-12 to calculate the energy consumption rate for the next grid point in RL'_k.

Algorithm 12: Pseudocode for GreedySearch Phase 3

Input: L, R_i, $\{RL_k\}$, G
Output: E^c_i

```
1  for ∀i ∈ RL_0 do
2  |   F_i = |Z(i)| × G
3  |   E^c_i = E^m + E^t F_i
4  end
5  for k = 1 ... K do
6  |   RL'_k = RL_k
7  |   while RL'_k ≠ ∅ do
8  |   |   i* = arg max_{i∈RL'_k} R_i
9  |   |   F_{i*} = F_{i*} + Σ_{h∈{RL_{k-1}∪N(i*)}} F_h
10 |   |   E^c_{i*} = E^c_{i*} + Σ_{h∈{RL_{k-1}∪N(i*)}} F_h E^t
11 |   |   RL_{k-1} = RL_{k-1} − RL_{k-1} ∪ N(i*)
12 |   |   RL'_k = RL'_k − {i*}
13 |   end
14 end
```

8.4 Analysis

This section presents properties of the proposed algorithms. It starts with the run time complexity of DirectSearch and GreedySearch. It then presents the upper bound of DirectSearch as compared to MILP in terms of required number of sensor nodes. After that, it shows GreedySearch is guaranteed to construct a connected network.

Proposition 13. *The run time complexity of DirectSearch is* $\mathcal{O}(n^2p^2m^2)$.

Proof. DirectSearch requires $|L|$ steps to calculate the weight of all grid points; see

Algorithm 9, *Line* 1-3. It then assigns the grid points into the sets RL and CL. This incurs $\mathcal{O}(|L|)$ iterations; see *Line* 4. In each iteration, there are three main steps. The calculation at *Line* 5 requires $|L|$ steps to determine the grid point with the minimum weight. In *Line* 6, which compares each element in $\{Q(i^*, P_j) \mid \forall j \in Z\}$ to the set $\{Q(i, P_j) \mid \forall j \in Z, \forall i \in L \cup CL \cup RL, i \neq i^*\}$, the size of both sets is respectively at most $p|Z|$ and $p|Z||L|$. Recall, p is the number of line segments. Therefore, the computational time of *Line* 6 is $\mathcal{O}(p^2|Z|^2|L|)$. *Line* 9 compares each element in the set $Z(i^*)$ to the set $\{Z(j) \mid \forall j \in L \cup CL, j \neq i^*\}$. This requires a maximal computational time of $\mathcal{O}(|Z| \times |Z|)$. Thus, the 'while' loop of *Line* 4-16 has a time complexity of $\mathcal{O}(|L|(|L| + p^2|Z|^2|L| + |Z|^2))$. Consequently, the total running time complexity of DirectSeach is $\mathcal{O}(|L|(|L| + p^2|Z|^2|L| + |Z|^2) + |L|)$. This can be revised to $\mathcal{O}(n^2p^2m^2)$ by substituting $|L| = n$ and $|Z| = m$. □

Proposition 14. *The run time complexity of GreedySearch is* $\mathcal{O}(\frac{4\pi r^2 n^3}{\lambda^2})$.

Proof. The run time complexity of GreedySearch is calculated respectively according to its three phases. In the first phase, it take $|L|$ steps to calculate the weight of all grid points; see Algorithm 10, *Line* 1-3. It then constructs the set CL, which needs a maximum of $|L|$ iterations; see Algorithm 10 *Line* 4. In each iteration, the main steps are determining the minimal weight grid point in L and determining whether the set $L(i^*) \cap Z$ is a subset of $\{L(i) \mid \forall i \in L \cup CL, i \neq i^*\}$; see Algorithm 10, *Line* 5 and *Line* 6. The running time of *Line* 5 is $\mathcal{O}(|L|)$. The running time of *Line* 6 is $\mathcal{O}(|L(i^*)||L|)$. This is because the maximum number of elements in the sets $\{L(i^*) \cap Z\}$ and $\{L(i) \mid \forall i \in L \cup CL, i \neq i^*\}$ are $|L(i^*)|$ and $|L|$, respectively. In order to calculate $\mathcal{O}(|L(i^*)|)$, which corresponds to the maximum number of integers within a circle with a radius of r, one can use the bound for the Gauss Circle Problem [83], which is $|L(i^*)| \leq \frac{\pi r^2}{\lambda^2} + \frac{2\sqrt{2}\pi r}{\lambda}$. Thus, the running time of Algorithm 10 *Line* 6 can be revised to $\mathcal{O}((\frac{\pi r^2}{\lambda^2} + \frac{2\sqrt{2}\pi r}{\lambda})|L|)$. Therefore, the run time complexity of GreedySearch phase one is $\mathcal{O}(|L| + (\frac{\pi r^2}{\lambda^2} + \frac{2\sqrt{2}\pi r}{\lambda} + 1)|L|^2)$. In the second phase, in the worst case, GreedySearch checks all the grid points in L. This means

the maximum number of iterations performed by the 'while' loops of Algorithm 11 *Line* 3 is $|L|$; i.e., $K = |L|$. In each iteration, it calls Algorithm 10 by replacing input r to r', where $r' = 2r$. Thus, the second phase of GreedySearch has a run time complexity of $\mathcal{O}(|L|^2 + (\frac{4\pi r^2}{\lambda^2} + \frac{4\sqrt{2}\pi r}{\lambda})|L|^3)$. In the last phase, GreedySearch performs K iterations; see Algorithm 12 *Line* 5. As shown earlier, K is equal to $|L|$. Moreover, the maximum number of elements in each RL_k is $|L|$. Therefore, the run time complexity of phase three is $\mathcal{O}(|L|^2)$. Consequently, the run time complexity of GreedySearch is $\mathcal{O}(|L| + (\frac{\pi r^2}{\lambda^2} + \frac{2\sqrt{2}\pi r}{\lambda} + 1)|L|^2 + |L|^2 + (\frac{4\pi r^2}{\lambda^2} + \frac{4\sqrt{2}\pi r}{\lambda})|L|^3 + |L|^2)$, which can be revised to $\mathcal{O}(\frac{4\pi r^2 n^3}{\lambda^2})$.

□

The next proposition presents the upper bound of DirectSearch as compared to MILP in terms of the required number of sensor nodes. Let R_{max} and R_{min} be respectively the maximum and minimum potential recharging rate in the sensing field, where $R_{max} \gg R_{min}$.

Proposition 15. *DirectSearch requires at most* $\frac{R_{max}}{R_{min}}$ *times more sensor nodes than MILP.*

Proof. Recall that DirectSearch only places sensor nodes on the grid points around the straight line connecting a target and the sink. Therefore, a large number of un-necessary sensor nodes will be deployed if all grid points around the straight line have a low potential recharging rate, i.e., R_{min}, especially if there are grid points away from the line with a higher recharging rate. This is precisely the case for MILP whereby it considers all grid points to deploy sensor nodes. Consequently, the optimal path calculated by MILP is one that uses grid points with a potential recharging rate of at most R_{max}. Let A and A' be the set of grid points on the path calculated by DirectSearch and MILP, respectively. As MILP is likely to use grid points with a high potential recharging rate which may deviate away from the path chosen by DirectSearch, it thus has $|A'| \geq |A|$. Let E_0 be the energy consumption rate incurred by sensor nodes to forward data. Therefore, the maximum ratio in

terms of the required number of sensor nodes calculated by DirectSearch over MILP is as follows,

$$\frac{\frac{|A|E_0}{R_{min}}}{\frac{|A'|E_0}{R_{max}}} = \frac{R_{max}|A|E_0}{R_{min}|A'|E_0} \leq \frac{R_{max}}{R_{min}}$$

□

Figure 8.4 shows an example of Proposition 15, whereby all grid points except 8 and 12 have a uniform potential recharging rate of R_{max}. There is one target located on the grid point 16. All sensor nodes have a sensing radius of $\lambda/2$. This means the target can only be monitored by sensor nodes on the grid point 16. Moreover, a sensor node can only communicate with others that located on its adjacent grid points. In this case, DirectSearch forwards data from 16 to the sink using the path 16-12-8-4. This is because the weight of all grid points except 8 and 12 are zero. Thus, the number of sensor nodes required by DirectSearch is $\frac{E^tG}{R_{max}} + \frac{2E^fG}{R_{min}}$. On the other hand, MILP use the path 16-15-11-7-3-4 to forward data which requires $\frac{E^tG}{R_{max}} + \frac{4E^fG}{R_{max}}$ sensor nodes in total. Thus, the ratio in this case is $\frac{R_{min}E^tG+2E^fGR_{max}}{R_{min}E^tG+4E^fGR_{min}} \approx \frac{R_{max}}{2R_{min}}$, which is less than $\frac{R_{max}}{R_{min}}$.

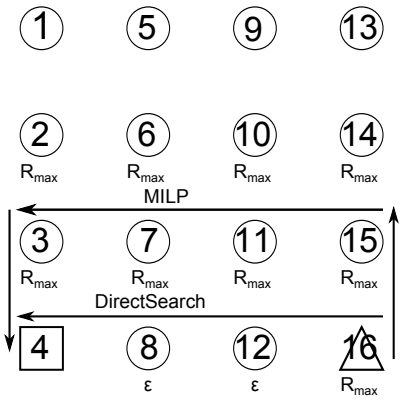


Figure 8.4: An example on DirectSearch and MILP determining a relay path from grid point 16 to 4. All sensor nodes have a sensing radius of $\lambda/2$ and a communication range of λ.

Proposition 16. *GreedySearch is guaranteed to construct a connected network.*

Proof. The first fact to show is that the grid points in CL, as constructed in phase one, are guaranteed to monitor all targets in Z. Suppose that there is a target $j \in Z$ that is not within the sensing range of all grid points in CL. This means GreedySearch has discarded all grid points in $L(j)$. This violates the rule whereby a grid point i is added into the set CL if any target, i.,e., j, is covered solely by i. Hence, all targets in Z must be covered by the grid points in CL.

Next is to show that the set of grid points in RL_k are guaranteed to connect to all grid points in RL_{k-1}. The second phase of GreedySearch is similar to the first phase. Instead, all nodes in RL_k are considered as "targets". This means all sensor nodes in RL_k must cover those nodes in RL_{k-1}. In other words, all nodes in RL_{k-1} have a connection to nodes in RL_k.

The next fact to be established is that the sink B has connectivity to all nodes in the set CL. Recall that, sink B is the only element in the set RL_K. This means it has connectivity to all nodes in RL_{K-1}. As mentioned earlier, as all nodes between relay sets are connected, this proof therefore concludes that B has connectivity to all nodes in CL, which completes the proof. □

8.5 Evaluation

In the evaluation, sensor nodes parameters are set as per the datasheet of WaspMote [72]. Specifically, all sensor nodes are equipped with an Enocean ECS310 solar cell [73], which has a conversion rate of 10% and recharging efficiency of 50%. This is a conservative assumption as compared to other technologies [74]. Further, the experiments use the recharging rate data collected from Southwest Solar Research Park, Phoenix, Arizona, USA [75] on the 16-th of April 2013. All parameter values used are summarized in Table 8.1.

The experiments used to evaluate the proposed algorithms comprise of "simple" and "complex" networks. In the simple network case, there are no more than four targets and the network dimension is less than 9λ. Specifically, two sets of experi-

Parameters	Value
Battery size	1100 mA [72]
Consumption rate	60mW [72]
Data generation rate	3.8 Kbytes/minute
Voltage	4V [72]
Solar panel conversion rate	10% [74]
Recharging efficiency	50% [74]
Average recharge rate	0.96 Joules/hour [73][75]
Transmission cost	0.1 Joules/Kbyte

Table 8.1: Simulation Parameters

ments are conducted in the simple network case by fixing the sensing radius to λ. First, the experiment investigates the impact of target density which is increasing from one to four with an interval of one where the network dimension is fixed at 5λ. In another set of experiments, the network dimension increases from 3λ to 9λ with an interval of λ and the number of targets is fixed to two. The evaluation only uses MILP in simple networks because it is computationally intractable for complex networks.

In the complex networks case, three set of experiments are conducted to further study the effect of target density and network dimension as well as sensing radius. The upper bound on the number of targets and network dimension is set to 50 and 25, respectively. In the first experiment, the number of targets is increasing from five to 50 with an interval of five with the sensing radius and network dimension fixed to 2λ and 15λ, respectively. After that, the second experiment studies the impact of network dimension ranging from 10λ to 25λ with an interval of λ with the number of targets set to 20 and sensing radius set to 3λ. In addition, in the last experiment, the number of targets and network dimension are fixed to 10 and 20λ respectively to observe the impact of sensing radius, which is increased from 2λ to 15λ with an interval of λ.

The experiments calculate the average value, out of 200 runs, of the following metrics:

- *Number of sensor nodes.* This is the number of sensor nodes required to achieve

energy neutral complete targets coverage and connectivity.

- *Number of sites.* This is the number of grid points that are required to place at least one sensor node to monitor targets or relay information.

- *Running time.* This is the computational time of Matlab 2014a on a computer with an Intel Core i7 CPU @ 3.5GHz with 8G of RAM.

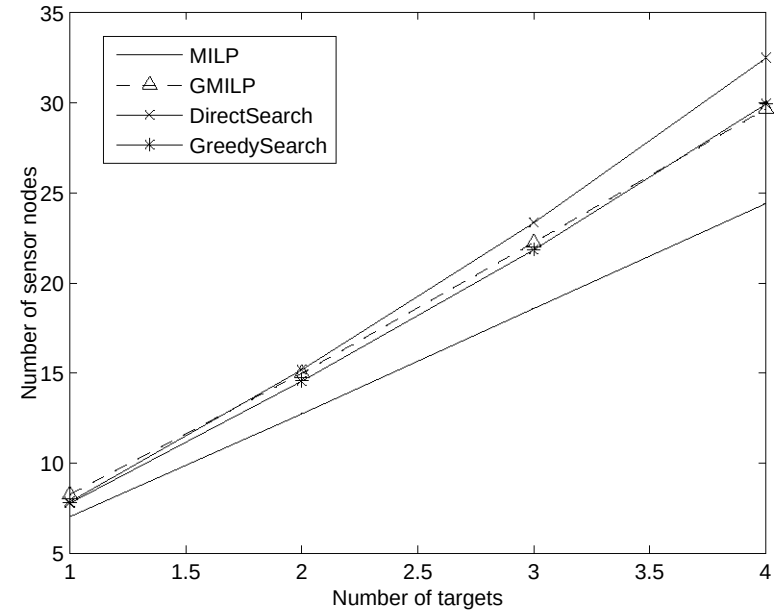
8.6 Results

8.6.1 Simple Networks

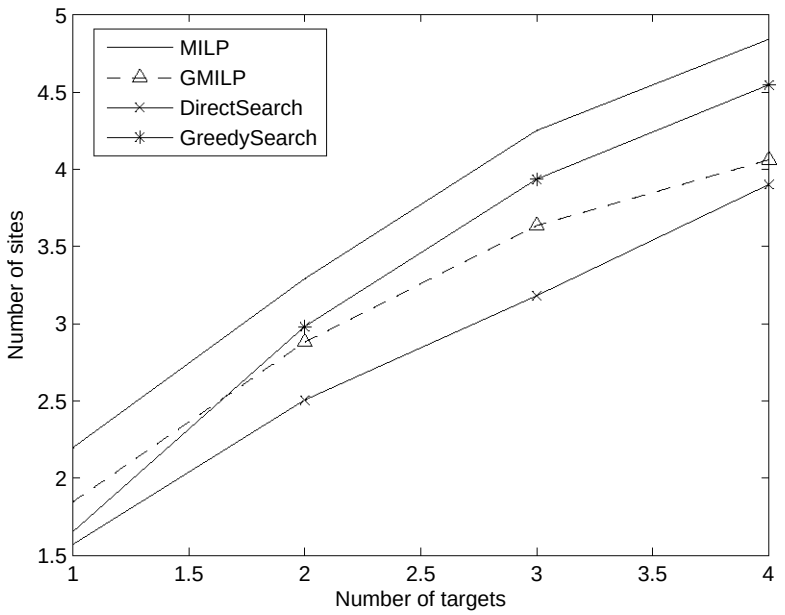
8.6.1.1 Target Density

From Figure 8.5a, it can be seen that the required number of sensor nodes for all algorithms to ensure energy neutral complete target coverage and connectivity increases with the addition of targets. This is because a higher number of targets require sensor nodes to expend more energy to monitor targets and forward data. We also observe that DirectSearch requires 20% more sensor nodes than MILP on average whilst this value is 10% for GreedySearch and GMILP. This is because DirectSearch only considers grid points that cover the straight line from targets to the sink. Hence, optimal points may not be selected by DirectSearch if their distance to the straight line is large.

From Figure 8.5b, it can be seen that the number of sites increases when more targets are in the sensing field. The reason is that additional sites are required to ensure coverage and connectivity for newly added targets. It also shows that MILP requires the highest number of sites amongst the four algorithms. This is because MILP considers all the grid points in the sensing field, and thus, sensor nodes can be deployed on various grid points to cooperatively monitor and relay data to minimize the total number of sensor nodes. However, the cooperation between the sensor nodes on different grid points increases the number of sites. Figure 8.5b shows



(a)



(b)

Figure 8.5: (a) Number of sensor nodes and (b) sites under different target densities

DirectSearch requires fewer sites than the other three algorithms. This is because the grid points considered by DirectSearch are around the straight line directly from targets to the sink. Thus, it requires fewer sites than other algorithms to ensure coverage and connectivity. Another observation is the number of sites required by GMILP is higher than GreedySearch when there is only one target. This is because the network dimension is small, and thus, GreedySearch can easily find a grid point directly connecting the sink to monitor the one target. However, the grid point selected by GMILP to monitor this target may not be directly connected to the sink. This means extra sites will be required to ensure connectivity. As can be seen that, GMILP requires fewer sites than GreedySearch when there are more than two targets.

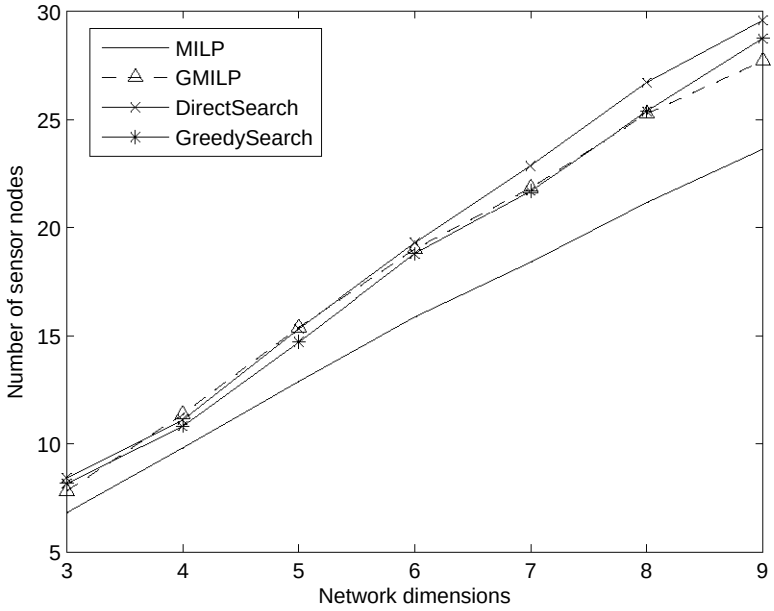
8.6.1.2 Network Dimension

From Figure 8.6a and 8.6b, it can be seen that the required number of sensor nodes and sites computed by MILP, GMILP, DirectSearch and GreedySearch increases with network dimension. This is because the average distance from targets to the sink increases with greater network dimensions. Consequently, more sites and relay nodes are required to ensure energy neutral connectivity. Figure 8.6a shows DirectSearch requires 20% more sensor nodes than MILP and 5% more sensor nodes than GMILP and GreedySearch on average. The reason is the same as that in Experiment 8.6.1.1 where DirectSearch only considers the grid points covering the straight line from targets to the sink.

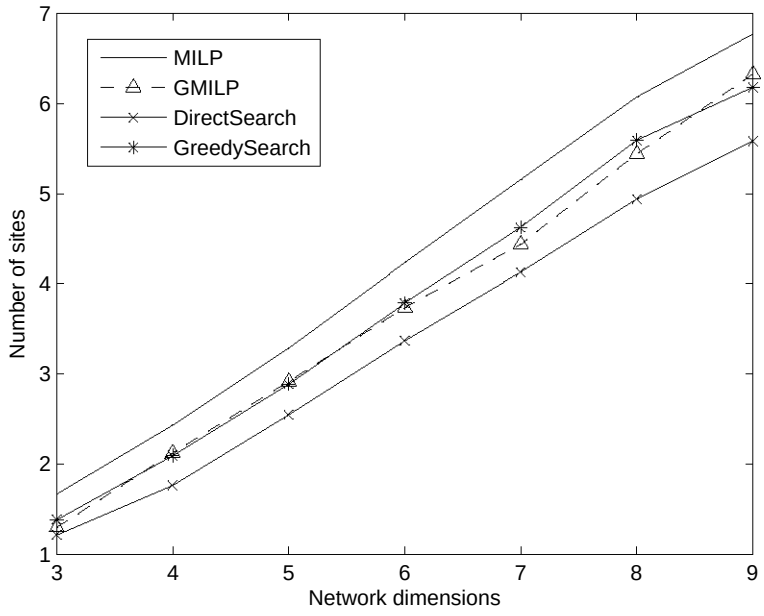
8.6.2 Complex Networks

8.6.2.1 Target Density

Figure 8.7a verifies the results in Section 8.6.1.1 where the number of sensor nodes required by GMILP, DirectSearch and GreedySearch increases with additional targets. The difference is that the curve for GreedySearch is steeper than GMILP.

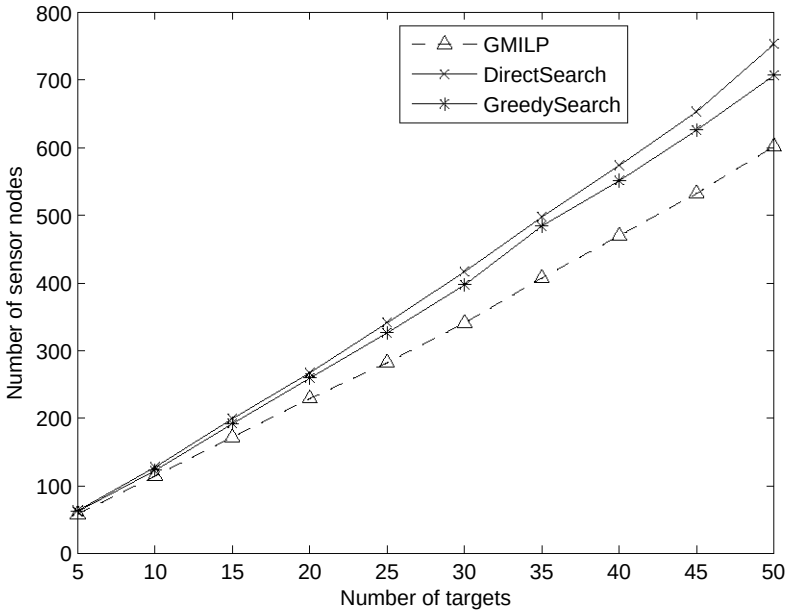


(a)

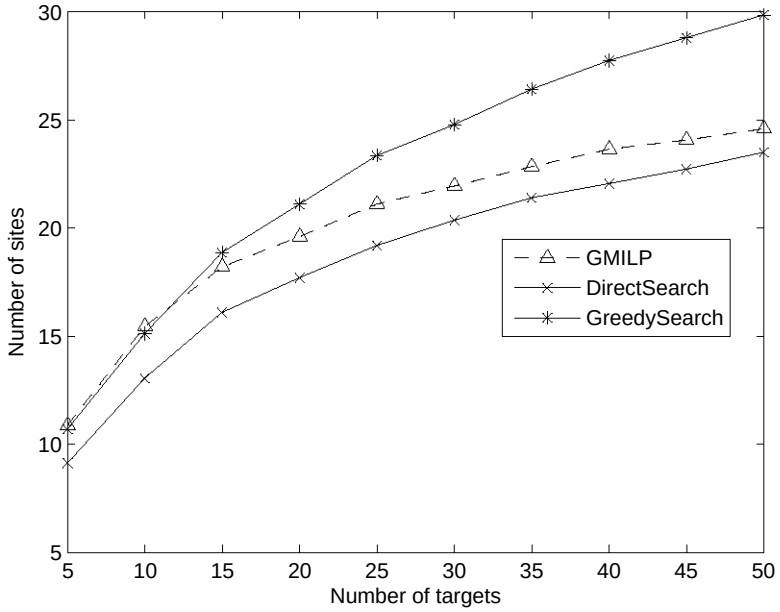


(b)

Figure 8.6: (a) Number of sensor nodes and (b) sites under different network dimensions



(a)



(b)

Figure 8.7: (a) Number of sensor nodes and (b) sites under different target densities

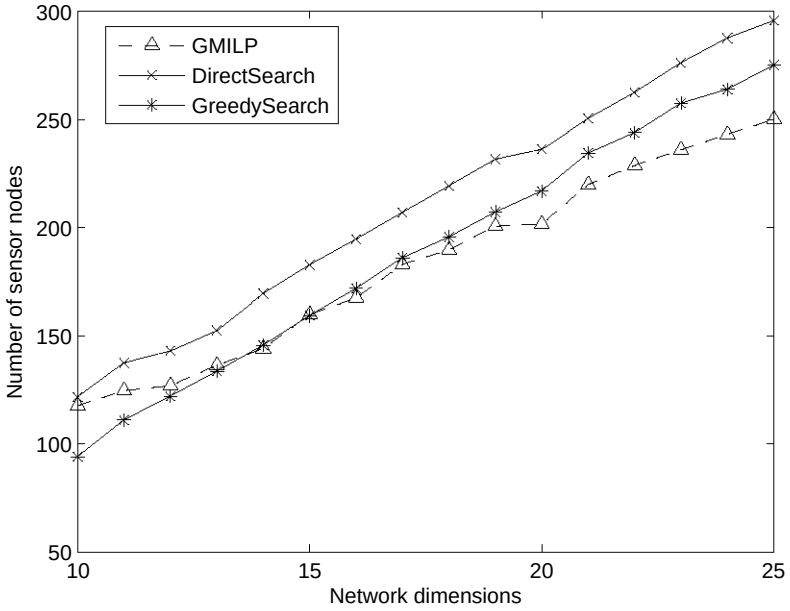
This is because GreedySearch greedily selects sites from targets to the sink that lead to unnecessary deployment of sensor nodes because these sites may have a low potential recharging rate. On the other hand, GMILP considers all candidate points and return the optimal solution using the MILP solver amongst these points.

Figure 8.7b shows the number of sites required by GMILP, DirectSearch and GreedySearch increases with more targets. However, the number of sites reaches a threshold when the number of targets is large. It can be seen that the additional number of sites required by GMILP, Direct and GreedySearch for five targets is less than 0.5 when there are 45 targets in the sensing field. This is because the selected sites provide complete area coverage; i.e., the entire sensing field is covered by these sites.

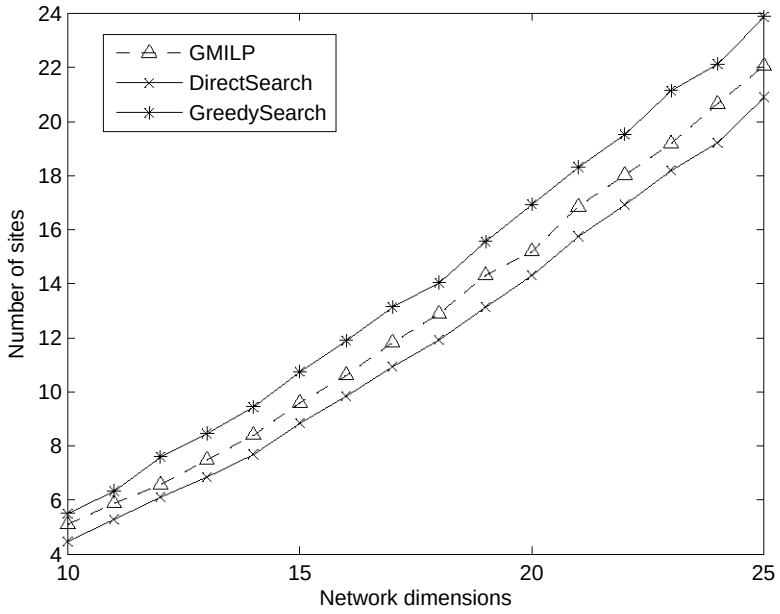
8.6.2.2 Network Dimension

Figure 8.8a shows the number of sensor nodes required by GMILP, DirectSearch and GreedySearch increases with network dimension. The reason is the same as that in Experiment 8.6.1.2 where additional sensor nodes are required to ensure energy neutral connectivity. Another observation is that the curve of GreedySearch and GMILP intersects when the network dimension is 15λ. This is because there are only a few candidate points when the network dimension is small. These candidate points, however, may not be ideal for sensor nodes to forward data. For example, GreedySearch can deploy sensor nodes on a straight line whilst GMILP sparsely place sensor nodes, which increases the total transmission cost.

Figure 8.8a confirms the results from Experiment 8.6.1.2. Moreover, it shows that the number of additional sites required by GMILP, DirectSearch and GreedySearch is similar. For example, they all require 1.6 more sites when the network dimension changed from 20λ to 21λ. This is because all the proposed algorithms search for grid points that efficiently forward sensed information to the sink. Thus, the number of additional sites required to place relay nodes is similar when the distance between targets and sink increases.



(a)



(b)

Figure 8.8: (a) Number of sensor nodes and (b) sites under different network dimensions

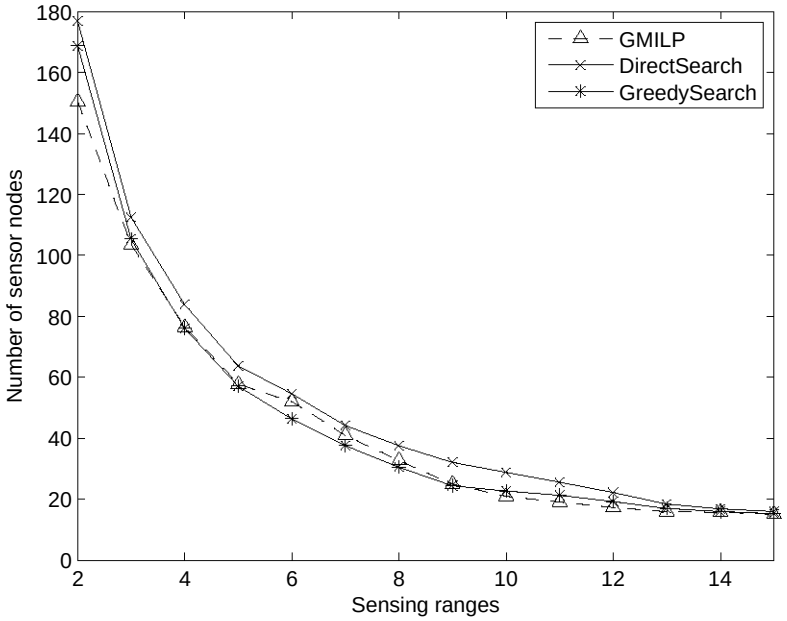
8.6.2.3 Sensing Radius

From Figure 8.9a and 8.9b, it can be seen that the number of sensor nodes and sites required by GMILP, DirectSearch and GreedySearch is reduced when increasing the sensing radius. The reason is because fewer sites and sensor nodes are required for relaying data when the sensing radius is large. Figure 8.9a also shows GMILP requires more sensor nodes than GreedySearch when the sensing radius is varied from 5λ to 9λ. The reason is because the number of candidate points computed by GMILP reduces with increasing sensing radius. For example, the number of candidate points is around 28 when the sensing radius is 3λ whilst this number is ten when the sensing radius is 6λ. Thus, the candidate points calculated by GMILP may have a low potential recharging rate. This causes unnecessary deployment of sensor nodes. However, GMILP outperforms GreedySearch in terms of the number of sensor nodes when increasing the sensing radius from ten to 13. This is because a few sites are sufficient to cover and relay data for these targets. For example, see Figure 8.9b, all three algorithms need 2.5 sites when sensing radius is 10λ. Thus, GMILP outperforms GreedySearch by two sensor nodes because it uses the MILP solver to determine the number of sensor nodes. When sensing radius is increased to 14λ, all three algorithms only need one site, see Figure 8.9b. In this case, the number of sensor nodes required by all the proposed algorithms is the same. This is because they all chose the grid point with the maximum potential recharging rate.

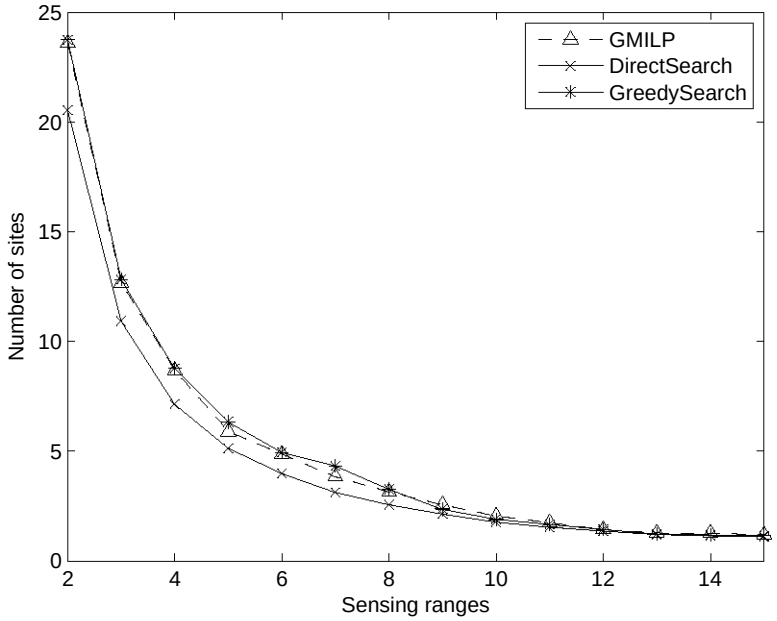
8.6.3 Running Time

This section first presents the run time result of MILP, GMILP, DirectSearch and GreedySearch in the simple networks case by varying the number of targets and network dimensions. It then shows the effect of targets densities, network dimensions and sensing radius on the running time of GMILP, DirectSearch and GreedySearch for the complex network case in Figure 8.10, 8.11 and 8.12.

Table 8.2 shows the running time when increasing the number of targets; i.e.,



(a)



(b)

Figure 8.9: (a) Number of sensor nodes and (b) sites under different sensing ranges

column T. It can be seen that the running time of MILP dramatically increases from seven seconds with three targets to 132 seconds with four targets in the sensing field. The reason is because the number of set covers in MILP has increased by more than 30 times with one additional target. The greater number of set covers increases the number of decision variables and computation complexity of MILP. Table 8.2 also shows that the running time of GMILP is significantly lower than MILP. This is because GMILP pre-selects candidate points to reduce the number of set covers, i.e., decision variables, in MILP.

T	**MILP**	**GMILP**	**DS**	**GS**
1	0.0317	0.0187	0.0037	0.0071
2	0.0762	0.0190	0.0057	0.0118
3	7.1188	0.0207	0.0087	0.0177
4	132.2321	0.0192	0.0104	0.0212

Table 8.2: Running time (seconds) with increasing target densities

Table 8.3 shows the running time when increasing the network dimension; i.e., column L. It can be seen that the running time of MILP rapidly increases from three to 153 seconds when the network dimension changes from 7λ to 9λ. The reason is the maximum number of connections in the sensing field has increased from 2401 to 6561 when the network dimension changes. The larger number of connections means more decision variables and computation time are required by the MILP solver. It also shows that the running time of GMILP, which pre-selects a set of candidate points to reduce the number of connections, is increased from 0.02 to 0.07 seconds when the network dimension changes from 7λ to 9λ.

The rest of this section presents results on the running time of GMILP, DirectSearch and GreedySearch in the complex networks case. Figure 8.10 shows that the running time of GMILP increases from 0.5 to 30 seconds. The reason is because the number of constraints in the MILP increases with additional targets; see Equation (8.4) and (8.5. Figure 8.11 shows the running time of GMILP rapidly increases from one to ten seconds when the network dimension changes from 20λ to

L	MILP	GMILP	DS	GS
3	0.0264	0.0169	0.0028	0.0045
4	0.0359	0.0177	0.0038	0.0074
5	0.0905	0.0187	0.0056	0.0111
6	0.4297	0.0225	0.0089	0.0188
7	3.2741	0.0243	0.0118	0.0264
8	9.5228	0.0284	0.0155	0.0343
9	153.2517	0.0650	0.0202	0.0430

Table 8.3: Running time (seconds) with increasing network dimensions

25λ. Moreover, Figure 8.12 shows the running time of GMILP drops from 75 to one second when the sensing radius increases from 2λ to 3λ. This is because the number of candidate points increases with network dimension whilst reduces with sensing radius. As shown in Section 8.3.1, the number of candidate points directly affects the number of decision variables and constraints in the GMILP algorithm. From all the three figures in Figure 8.10, 8.11 and 8.12, it can be seen that GreedySearch requires more running time than DirectSearch. This is because DirectSearch only calculates the weight of all grid points upon initialization whilst GreedySearch needs to re-compute the weight of all grid points after each iteration.

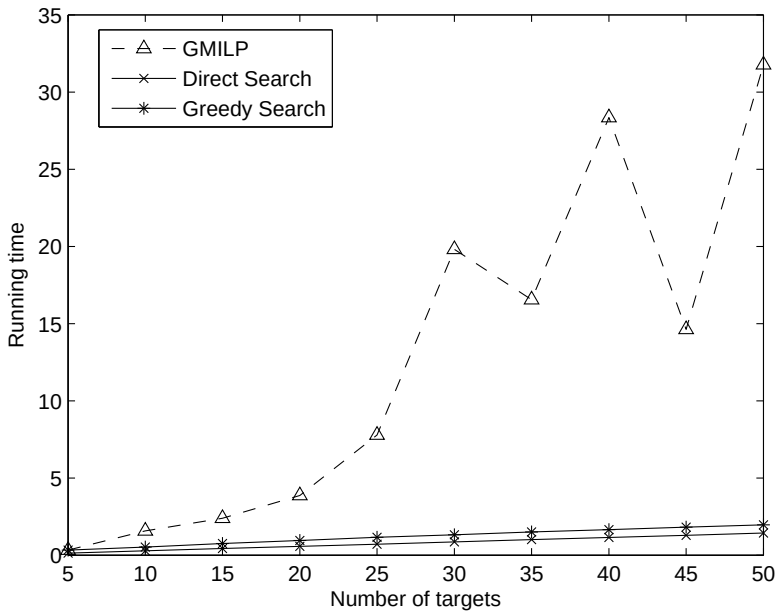


Figure 8.10: Running time under different target densities

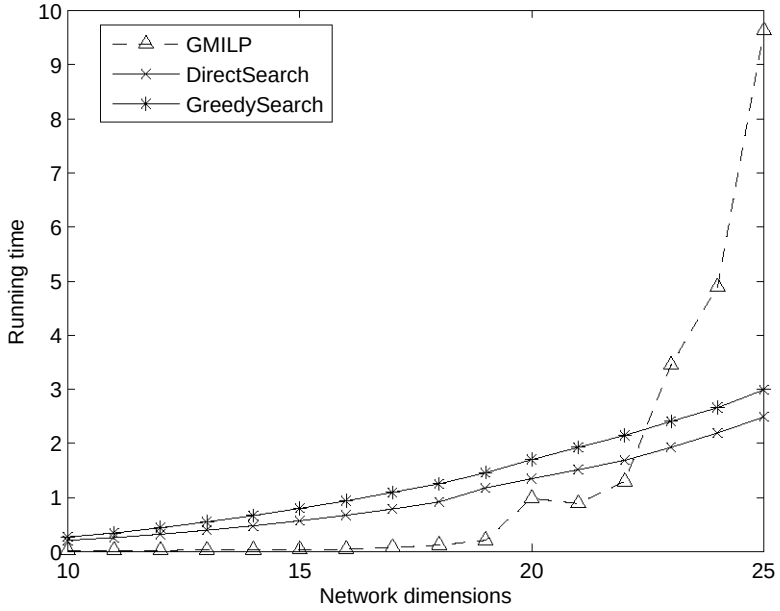


Figure 8.11: Running time under different network dimensions

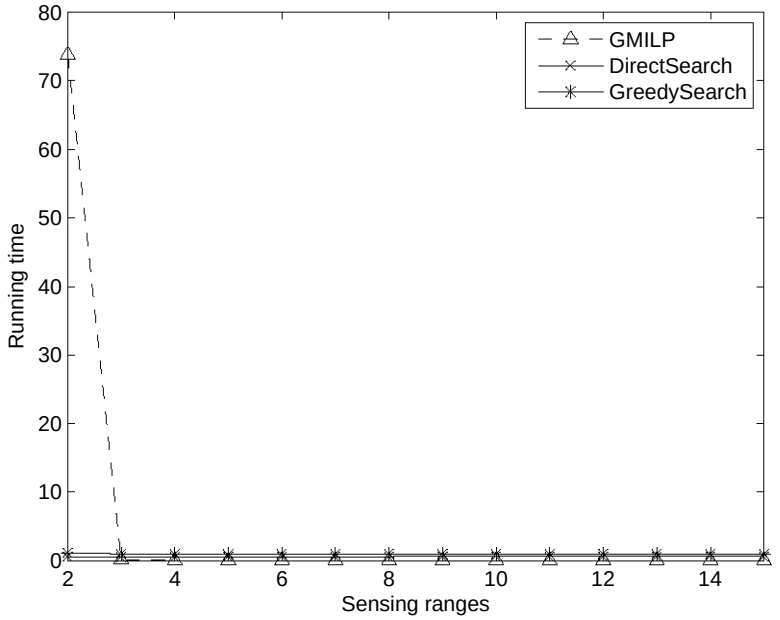


Figure 8.12: Running time under different sensing ranges

8.7 Conclusion

This chapter presents the first work that jointly considers nodes placement and duty cycling to ensure complete targets coverage, network connectivity and energy neutral operation. To address the MEHNP-ENCC problem, both optimal and heuristic solutions are proposed with varying level of computational complexity. The first approach involves solving a MILP with an objective to minimize sensor nodes subject to complete targets coverage, energy and network connectivity constraints. The MILP, however, requires an exhaustive collection of set covers. Therefore, GMILP is proposed to reduce the number of set covers or decision variables considered by the MILP. In addition, this chapter contains two heuristics: DirectSearch and GreedySearch. These heuristics iteratively determine the locations to place sensing and relaying nodes. Simulation results show that the DirectSearch needs to place 20% more sensor nodes than MILP whilst GreedySearch and GMILP only require 10%. They also show that the running time of DirectSearch and GreedySearch is an order faster than GMILP and four orders faster than MILP.

Chapter 9

Conclusion

Complete targets coverage is a key requirement in many surveillance applications that rely on WSNs. It ensures that all events that occur at a target location will be recorded. To date, existing works only study the duty cycling for complete targets coverage problem in conventional or non-rechargeable WSNs. This means that the operation time of a network is limited by the limited battery capacity of sensor nodes. They also assume that a central controller knows the exact battery information at sensor nodes. Furthermore, past works on energy harvesting WSNs have only focused on maximizing targets detection probability. In other words, they do not require the targets to be monitored at all times. Another key assumption of past duty cycling algorithms is that sensor nodes are already deployed around targets. They thus do not guarantee energy neutral operation.

In light of the aforementioned gaps in current state-of-the-art, this thesis has addressed the following research questions:

1. How to effectively schedule the activation time of sensor nodes to maximize network lifetime whilst maximizing harvested energy to ensure complete targets coverage? This question further considers three scenarios: (1) what if the base station/sink does not know the accurate battery level of sensor nodes? (2) what if sensor nodes can make decision by themselves? and (3) what if

sensor nodes are not able to connect to the base station/sink directly?

2. How to deploy energy harvesting sensor nodes to completely cover all targets and achieve energy neutral operation? Moreover, how to place these sensor nodes such that they are connected to a sink?

To address the first question, Chapter 3 proposes the MLCEH subproblem and outlines two novel solutions, called LP-MLCEH and MUA. The LP-MLCEH solution addresses the MLCEH problem using a LP solver. On the other hand, MUA is a heuristic algorithm that selects sensor nodes with a high residual energy to monitor target(s). A key consideration is that a sensor node will be activated immediately when it has a full battery. This is to conserve the recharging opportunities of sensor nodes. Chapter 3 also shows via simulation that LP-MLCEH is able to double network lifetime as compared to similar algorithms developed for finite battery WSNs in the context of energy harvesting. It also shows that MUA achieves 3/4 of the network lifetime obtained by LP-MLCEH and do so with far less computation time. Chapter 4 addresses the MLCEH problem in scenarios where the base station/sink does not have the up to date knowledge of the current battery level of sensor nodes. It contains a two stage SP [71] based solution and also relies on the SAA framework [28]. The resulting algorithm, called SP-UMLC, achieves 80% of the theoretically achievable network lifetime. After that, Chapter 5 outlines a distributed algorithm, i.e., MEP, for the MLCEH problem. It allows a sensor node to turn itself on/off based on two-hops information. Chapter 5 compares MEP to two distributed algorithms developed for finite battery WSNs in the context of energy harvesting WSNs. Results show that MEP increases network lifetime by at least 30% and yields lower coverage redundancy. To address the last scenario of the first question, Chapter 6 considers the MLCCEH problem and proposes LP-MLCCEH and EC-MLCCEH; these two algorithms ensure all activated sensor nodes maintain complete targets coverage and have path(s) to forward data to the base station/sink. The LP-MLCCEH formulates the MLCCEH problem as a LP with an

objective to maximize network lifetime subject to coverage, energy and flow conservation. On the other hand, EC-MLCCEH iteratively selects sensor nodes with a high residual energy to monitor targets and forwards sensed data to the sink. Simulation results show that EC-MLCCEH achieves 80% network lifetime as computed by LP-MLCCEH at a fraction of the computation time.

Chapter 7 and 8 address the second research question. Specifically, Chapter 7 considers the MEHNP-PC problem whilst Chapter 8 addresses the MEHNP-ENCC problem which considers network connectivity. Chapter 7 proposes three approximation algorithms, i.e., GRNP, TPNP and EENP, to address the MEHNP-PC problem. The objective of these algorithms is to determine the location of sensor nodes such that the network has energy neutral operation. These algorithms first round down the fractional solution derived from the ILP for the MEHNP-PC problem. This, however, yields an infeasible solution. They then use different policies to construct good feasible solutions. Chapter 7 also proves that the EENP algorithm has the lowest approximation ratio which is $\frac{1}{2}|Z| + \frac{3}{2}$. Futhermore, simulation results show that EENP is close to optimal with less than 1% gap in terms of the required number of sensor nodes. On the other hand, Chapter 8 first formulates the MEHNP-ENCC problem as a MILP. Its objective is to minimize the deployed number of sensor nodes subject to energy neutral operation, complete targets coverage and connectivity constraints. The main input is all the combinations of locations to place sensor nodes. However, this is computationally intractable. Consequently, a greedy heuristic called GMILP is proposed to reduce the number of decision variables considered by the MILP. Further, Chapter 8 proposes two heuristics, i.e., DirectSearch and GreedySearch, to iteratively determine the location to deploy sensor nodes for sensing and relaying. Simulation results show that DirectSearch needs to place 20% more sensor nodes than MILP whilst GreedySearch and GMILP only require 10%. Moreover, the running time of DirectSearch and GreedySearch is an order faster than GMILP and four orders faster than MILP.

A key future research direction is to investigate mobile sensor nodes whereby

they are able to re-locate themselves to satisfy a specific requirement. For example, sensor nodes can be moved to maintain complete targets coverage when activated sensor nodes fail unexpectedly. This thus saves the cost of replacing failed nodes. Another research direction is to recharge sensor nodes by using a mobile charging router, which can be a robot or an unmanned vehicle. Then the research question becomes how to plan the trajectory of one or more routers/vehicles to ensure all sensor nodes have a positive battery level.

Bibliography

[1] P. H. C. Chulsung Park, "Ambimax: Autonomous energy harvesting platform for multi-supply wireless sensor nodes," in *3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, (Reston, USA), September 2006.

[2] K. S. P. Brett Warneke, Bryan Atwood, "Smart dust mote forerunners," in *14th IEEE International Conference on Micro Electro Mechanical Systems*, (Interlaken, Switzerland), January 2001.

[3] T. Starner, "Human-powered wearable computing," *IBM Systems Journal*, vol. 35, pp. 618–629, 1996.

[4] D. C. Jay Taneja, Jaein Jeong, "Design, modeling, and capacity planning for micro-solar power sensor networks," in *7th International Conference on Information Processing in Sensor Networks*, (St. Louis, USA), April 2008.

[5] H. Salarian, K.-W. Chin, and F. Naghdy, "Coordination in wireless sensor actuator networks," *Elsevier Journal on Parallel and Distributed Computing*, vol. 7, no. 72, pp. 856–867, 2012.

[6] S. Ganeriwal, I. Tsigkogiannis, H. Shim, V. Tsiatsis, M. B. Srivastava, and D. Ganesan, "Estimating clock uncertainty for efficient duty-cycling in sensor

networks," *IEEE/ACM Transactions on Networking*, vol. 17, pp. 843–856, June 2009.

[7] H. Liu, A. Chandra, and J. Srivastava, "esense: energy efficient stochastic sensing framework for wireless sensor platforms," in *Fifth International Conference on Information Processing in Sensor Netowrks*, (Nashville, USA), April 2006.

[8] J. Paek, B. Greenstein, O. Gnawali, K.-Y. Jang, A. Joki, M. Vieira, J. Hicks, D. Estrin, R. Govindan, and E. Kohler, "The tenet architecture for tiered sensor networks," *ACM Transactions on Sensor Networks*, vol. 6, pp. 34: 1–44, July 2010. Issue 4.

[9] X. Wang, G. Xing, Y. Zhang, C. L. R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *The First ACM Conference on Embedded Networked*, (Los Angeles, USA), November 2003.

[10] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol forwireless microsensor networks," in *33rd Annual Hawaii International Conference on System Sciences*, (Island of Maui), Jan 2000.

[11] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: survey and implications," *IEEE Communications Surveys & Tutorials*, vol. 13, pp. 443 – 461, Sept 2011.

[12] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi, "Hardware design experiences in zebranet," in *2nd International Conference on Embedded Network Sensor Systems*, (Baltimore, USA), November 2004.

[13] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, *et al.*, "A macroscope in the redwoods," in *3rd International Conference on Embedded Network Sensor Systems*, (San Diego, USA), November 2005.

[14] J. H. Huang, S. Amjad, and S. Mishra, "Cenwits: a sensor-based loosely coupled search and rescue system using witnesses," in *3rd International Conference on Embedded Networked Sensor Systems*, (San Diego, USA), November 2005.

[15] G. Simon, M. Maróti, Á. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton, "Sensor network-based countersniper system," in *2nd International Conference on Embedded Network Sensor Systems*, (Baltimore, USA), November 2004.

[16] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing*, vol. 10, no. 2, pp. 18–25, 2006.

[17] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P.Corke, "Data collection, storage, and retrieval with an underwater sensor network," in *3rd International Conference on Embedded Networked Sensor Systems*, (San Diego, USA), November 2005.

[18] J. Sorber, A. Kostadinov, M. Garber, M. Brennan, M. D. Corner, and E. D. Berger, "Eon: a language and runtime system for perpetual systems," in *5th International Conference on Embedded Networked Sensor Systems*, (Sydney, Australia), November 2007.

[19] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler, "Trio -enabling sustainable and scalable outdoor wireless sensor network deployment," in *The Fifth International Conference on Information Processing in Sensor Network*, (Nashville, USA), April 2006.

[20] J. A. Paradiso and M. Feldmeier, "A compact, wireless, self-powered pushbutton controller," in *3rd International Conference on Ubiquitous Computing*, (Atlanta, USA), Springer-Verlag, September 2001.

[21] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and

M. Srivastava, "Cyclops: in situ image sensing and interpretation in wireless sensor networks," in *3rd International Conference on Embedded Networked Sensor Systems*, (San Diego, USA), November 2005.

[22] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis, "Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea," in *3rd International Conference on Embedded Networked Sensor Systems*, (San Diego, USA), November 2005.

[23] X. Jiang, J. Polastre, and D. Culler, "Perpetual environmentally powered sensor networks," in *Fourth International Symposium on Information Processing in Sensor Networks*, (Los Angeles, USA), April 2005.

[24] S. Chalasani and J. M. Conrad, "A survey of energy harvesting sources for embedded systems," in *South East Conference*, (Huntsville, USA), April 2008.

[25] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, pp. 1–38, Sept 2007.

[26] K. Kar and N. Jaggi, "Dynamic node activation in networks of rechargeable sensors," *IEEE/ACM Transactions on Networking*, vol. 14, pp. 15–26, February 2006.

[27] B. Wang, *Coverage Control in Sensor Networks*, ch. Coverage Lifetime Maximization, pp. 65–95. Springer, 2010.

[28] W.-K. Mak, D. P. Morton, and R. K. Wood, "Monte carlo bounding techniques for determining solution quality in stochastic programs," *Operations Research Letters*, vol. 24, pp. 47–56, 1999.

[29] M. X. Cheng and X. Gong, "Maximum lifetime coverage preserving scheduling

algorithm in sensor networks," *Journal of Global Optimization*, vol. 51, pp. 447–462, 2011.

[30] M. Cardei, M. T. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in *IEEE INFOCOM*, (Miami, USA), Mar 2005.

[31] N. Ahn and S. Park, "An optimization algorithm for the maximum lifetime coverage pproblem in wireless sensor network," *International Journal of Management Science*, vol. 17, no. 2, pp. 1–23, 2011.

[32] S.-Y. Pyun and D.-H. Cho, "Power-saving scheduling for multiple-target coverage in wireless sensor networks," *IEEE Communications Letters*, vol. 13, no. 2, pp. 130–132, 2009.

[33] M. Cardei and D.-Z. Du, "Improving wireless sensor network lifetime through power aware organization," *Wireless Networks*, vol. 11, pp. 333–340, May 2005.

[34] P. BerMan, G. Calinescu, C. Shah, and A. Zelikovsky, "Power efficient monitoring management in sensor networks," *Wireless Communication and Networking Conference*, vol. 4, pp. 2329–2334, March 2004.

[35] N. Ahn and S. Park, "A new mathematical formulation and a heuristic for the maximum disjoint set covers problem to improve the lifetime of the wireless sensor network," *Ad Hoc & Sensor Wireless Networks*, vol. 13, pp. 209–225, 2011.

[36] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in *IEEE ICC*, 2001.

[37] B. Wang, "Coverage problems in sensor networks: A survey," *ACM Computing Surveys (CSUR)*, vol. 43, pp. 1–53, Oct 2011.

[38] N. Garg and J. Konemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," in *Proceedings of FOCS*, 1997.

[39] A. Pigatti, M. P. de Aragao, and E. Uchoa, "Stabilized branch-and-cut-and-price for the generalized assignment problem," in *Electronic Notes in Discrete Mathematics*, 2005.

[40] D. Tian and N. D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 3, no. 2, pp. 271–290, 2003.

[41] D. Brinza and A. Zelikovsky, "Deeps: Deterministic energy-efficient protocol for sensor networks," in *7th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp. 261–266, June 2006.

[42] M. N. Islam, Y. M. Jang, S. Choi, S. Park, and H. Park, "Redundancy reduction protocol with sensing coverage assurance in distributed wireless sensor networks," in *9th International Symposium on Communications and Information Technology*, pp. 631–636, 2009.

[43] L. Gu and J. A. Stankovic, "Radio-triggered wake-up capability for sensor networks," *Real-Time System*, vol. 29, pp. 157–182, March 2005.

[44] T. He, P. Vicaire, T. Yan, L. Luo, L. Gu, G. Zhou, R. Stoleru, Q. Cao, J. A. Stankovic, and T. Abdelzaher, "Achieving real-time target tracking using wireless sensor networks," in *Twelfth IEEE Real-Time and Embedded Technology and Application System*, 2006.

[45] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang, "Peas: A robust energy conserving protocol for long-lived sensor networks," in *23th International Conference on Distributed Computing System*, pp. 28–37, 2003.

[46] T. Yan, T. He, and J. A. Stankovic, "Differentiated surveillance for sensor networks," in *1st International conference on Embedded Networked sensor system*, pp. 51–62, 2003.

[47] S. K. Prasad and A. Dhawan, "Distributed aalgorithm for lifetime of wireless sensor networks based on dependencies among cover sets," in *14th International Conference on High Performance Computing*, 2007.

[48] H. Zhang, H. Wang, and H. Feng, "A distributed optimum algorithm for target coverage in wireless sensor networks," *Asia-Pacific Conference on Information Processing*, vol. 2, pp. 144–147, 2009.

[49] H. Liu, X. Jia, P.-J. Wan, C.-W. Yi, S. K. Makki, and N. Pissinou, "Maximizing lifetime of sensor surveillance system," *IEEE/ACM Transactions on Networking*, vol. 15, no. 2, pp. 334–344, 2007.

[50] H. Liu, X. Chu, Y.-W. Leung, X. Jia, and P.-J. Wan, "General maximal lifetime sensor-target surveillance problem and its solution," *IEEE transactions on Parallel and Distributed Systems*, vol. 22, no. 10, pp. 1757–1765, 2011.

[51] T. Zhao and Q. Zhao, "Lifetime maximization based on coverage and connectivity in wireless sensor networks," *Journal of Signal Processing System*, vol. 57, no. 3, pp. 385–400, 2009.

[52] A. Alfieri, A. Bianco, P. Brandimarte, and C. F. Chiasserini, "Maximizing system lifetime in wireless sensor networks," *European Journal of Operational Research*, vol. 181, no. 1, pp. 390–402, 2007.

[53] V. Pryyma, D. Turgut, and L. Bölöni, "Active time scheduling for rechargeable sensor networks," *Computer Networks*, vol. 54, no. 4, pp. 631–640, 2010.

[54] Z. Ren, P. Cheng, J. Chen, D. K. Y. Yau, and Y. Sun, "Dynamic activation policies for event capture with rechargeable sensors," in *32nd IEEE International Conference on Distributed Computing System*, (Macau, China), June 2012.

[55] N. Jaggi, K. Kar, and A. Krishnamurthy, "Rechargeable sensor activation under temporally correlated event," *Wireless Networks*, vol. 15, pp. 619–635, July 2009.

[56] M. L. Puterman, "Markov decision processes: Discrete dynamic stochastic programming," 1994.

[57] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman, "Acting optimally in partially observable stochastic domains," in *12th National Conference on Artificial Intelligence*, vol. 2, pp. 1023–1028, JOHN WILEY & SONS LTD, 1995.

[58] T. Banerjee and A. A. Kherani, "Sensor node activation policies using partial or no information," in *IEEE WiOpt*, (Limassol, Cyprus), Apr. 2007.

[59] S. Tang, X. Li, X. Shen, J. Zhang, G. Dai, and S. K. Das, "Cool: On coverage with solar-powered sensors," in *31st International Conference on Distributed Computing System*, 2011.

[60] M. Patel, R. Chandrasekaran, and S. Venkatesan, "Energy efficient sensor, relay and base station placements for coverage, connectivity and routing," in *24th IEEE International Performance, Computing, and Communications Conference*, (Phoenix, USA), Apr 2005.

[61] P. Cheng, C.-N. Chuah, and X. Liu, "Energy-aware node placement in wireless sensor networks," in *IEEE Global Telecommunications Conference*, (Texas, USA), Dec 2004.

[62] H. Liu, X. Chu, Y.-W. Leung, and R. Du, "Minimum-cost sensor placement for required lifetime in wireless sensor-target surveillance networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, pp. 1783–1796, Jul 2013.

[63] S. Dhillon and K. Chakrabarty, "Sensor placement for effective coverage and surveillance in distributed sensor networks," in *IEEE Wireless Communications and Networking*, (New Orleans), March 2003.

[64] X. Xu and S. Sahni, "Approximation algorithms for sensor deployment," *IEEE Transactions on Computers*, vol. 56, pp. 1681–1695, Dec 2007.

[65] Q. Wu, N. S. Rao, X. Du, S. S. Iyengar, and V. K. Vaishnavi, "On efficient deployment of sensors on planar grid," *Computer Communications*, vol. 30, pp. 2721–2734, Oct 2007.

[66] Z. A. Eu, H.-P. Tan, and W. K. G. Seah, "Routing and relay node placement in wireless sensor networks powered by ambient energy harvesting," in *IEEE Wireless Communications and Networking Conference*, (Budapest, Hungary), Apr 2009.

[67] S. Misra, N. Majd, and H. Huang, "Approximation algorithms for constrained relay node placement in energy harvesting wireless sensor networks," *IEEE Transactions on Computers*, pp. 1–14, Aug 2013.

[68] S. Misra, N. E. Majd, and H. Huang, "Constrained relay node placement in energy harvesting wireless sensor networks," in *8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, (Valencia, Spain), Oct 2011.

[69] S. He, J. Chen, F. Jiang, D. Yau, G. Xing, and Y. Sun, "Energy provisionin in wireless rechargeable sensor networks," *IEEE Transactions on Mobile Computing*, vol. 12, pp. 1931 – 1942, Aug. 2013.

[70] L. Xie, S. Yi, T. Hou, W. Lou, H. Sherali, and S. Midkiff, "On renewable sensor networks with wireless energy transfer: The multi-node case," in *IEE SECON*, (Seoul, South Korea), June 2012.

[71] D. Birge and F. Louveaux, *Introduction to Stochastic Programming*. Springer, 1997.

[72] "Waspmote datasheet." http://www.libelium.com/downloads/documentation/waspmote_datasheet.pdf.

[73] "Enocean ECS 310." http://www.enocean.com.

[74] A. C. Valera, W.-S. Soh, and H.-P. Tan, "Energy-neutral scheduling and forwarding in environmentally-powered wireless sensor networks," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1202–1220, 2013.

[75] "Nrel: Southwest solar research park." http://www.nrel.gov/midc/ssrp/.

[76] C. Yang and K. Chin, "Novel algorithms for complete targets coverage in energy harvesting wireless sensor networks," *IEEE Communications Letters*, vol. 18, pp. 118–121, Jan 2014.

[77] M. Rahimi, H. Shah, G. Sukhatme, J. Heideman, and D. Estrin, "Studying the feasibility of energy harvesting in a mobile sensor network," in *IEEE International Conference on Robotics and Automation*, (Taipei), Sept 2003.

[78] C. Yang and K. Chin, "A novel distributed algorithm for complete targets coverage in energy harvesting wireless sensor networks," in *IEEE International Conference on Communications*, (Sydney, Australia), June 2014.

[79] J. Y. Yen, "Finding the k shortest loopless paths in a network," *management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[80] C. Zhu, C. Zheng, L. Shu, and G. Han, "A survey on coverage and connectivity issues in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 35, pp. 619–632, Mar 2012.

[81] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of operations research*, vol. 4, no. 3, pp. 233–235, 1979.

[82] S. T. L. Heath, *A history of Greek mathematics*, vol. 2. Oxford, 1921.

[83] G. H. Hardy, *Ramanujan: Twelve lectures on subjects suggested by his life and work*. New York: Chelsea, 1999.