OPTIMAL PLACEMENT, SCHEDULING AND ROUTING TO MAXIMIZE LIFETIME IN WIRELESS SENSOR NETWORKS UNDER CONNECTIVITY RESTRICTIONS

by

Banu Kabakulak

B.S. in Industrial Engineering, Boğaziçi University, 2007

B.S. in Mathematics, Boğaziçi University, 2007

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2010

OPTIMAL PLACEMENT, SCHEDULING AND ROUTING TO MAXIMIZE LIFETIME IN WIRELESS SENSOR NETWORKS UNDER CONNECTIVITY RESTRICTIONS

APPROVED BY:

Prof. İ. Kuban Altınel (Thesis Supervisor)	.....
Assoc. Prof. M. Necati Aras	.....
Prof. Cem Ersoy	.....

DATE OF APPROVAL: 15.09.2010

to my family,
Mr. Ergün Önal
and Prof. İ. Kuban Altınel

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my thesis supervisor Prof. İ. Kuban Altınel, for his wise, enlightening ideas, invaluable contributions, guidance and endless motivation in this study. It has been a real pleasure to work with him in this work. I also would like to thank Assoc. Prof. M. Necati Aras for valuable guidance throughout this project. I am grateful to Prof. Cem Ersoy for his interest, comments and willingness to serve as my commmittee member.

I would like to thank all of my collegues for their technical support and inspring discussions during the development of solution strategies of the model. Specifically, I thank Yavuz Boğaç Türkoğulları for sharing details of his Ph.D. thesis with me.

Finally, I also feel grateful towards my family and my mathematics teacher Mr. Ergün Önal at high school for their great supports and valuable trust during all my life and also in my education.

ABSTRACT

OPTIMAL PLACEMENT, SCHEDULING AND ROUTING TO MAXIMIZE LIFETIME IN WIRELESS SENSOR NETWORKS UNDER CONNECTIVITY RESTRICTIONS

A wireless sensor network consists of distributed autonomous electronic devices called sensors. They are capable of sensing the changes in their vicinity, process the information as data packets and transmit the data to other sensors or a base station namely sink. In order to have an effective sensor network that can keep track of the changes in the interested region, sensors have to work cooperatively since they have limited battery energy. Working in accordance is also important to transmit the collected information eventually to a sink, since sensors can communicate only with the others that fall in a certain range. In most of the real life applications, for a wireless sensor network the number of periods that the network can operate as desired is a significant performance indicator.

In this thesis, we propose mixed-integer linear programming models to maximize the network lifetime by optimally determining the locations of sensors, activity schedules of the deployed sensors, sink assignments of the active sensors and their data flow routes to the corresponding sink over a finite planning horizon subject to coverage, flow conservation, energy consumption and budget constraints. Then, we introduce valid inequalities to solve the problem easily. Due to the characteristics of the problem, even the small instances cannot be solved exactly in considerable amount of time and the linear programming relaxations give poor upper bounds. Hence, we develop heuristics using techniques such as Lagrangean relaxation and greedy selection criterion. Computational experiments indicate that the heuristic methods are accurate and efficient.

ÖZET

KABLOSUZ DUYGAÇ AĞLARININ ÖMRÜNÜ EN BÜYÜKLEMEK İÇİN YERLEŞTİRME, ÇİZELGELEME VE ROTALAMA PROBLEMLERİNİN BAĞLILIK KISITLARI ALTINDA ÇÖZÜLMESİ

Bir kablosuz duygaç ağı, duygaç adı verilen, dağıtık ve bağımsız çalışabilen elektronik aygıtlardan oluşur. Duygaçlar yakınlarında meydana gelen değişiklikleri duyumsayabilir, bu bilgiyi veri paketi olarak işleyebilir ve verileri diğer duygaçlara ya da ana alıcılara iletebilir. İlgilenilen bölge ile ilgili değişiklikleri takip edebilen etkin bir duygaç ağı oluşturabilmek için kısıtlı pil enerjisine sahip duygaçların birbirleriyle uyumlu çalışması gerekmektedir. Uyumlu calışma, duygaçlar sadece belli bir aralıkta yer alan duygaçlarla iletişim kurabildiğinden, toplanan bilgilerin sonunda bir alıcıya iletilebilmesi için de önemlidir. Uygulamaların çoğunda, bir kablosuz duygaç ağı için ağın istenildiği gibi çalışabildiği dönem sayısı anlamlı bir başarı göstergesidir.

Bu tezde, ilkin duygaçların en iyi yerlerini, en iyi etkinlik çizelgelerini, çalışan duygaçların alıcı atamalarını ve duygaçlardan alıcılarına olan en iyi bilgi akış rotalarını bularak ağın ömrünü enbüyükleyen karışık tamsayılı programlama gösterimleri geliştirilmektedir. Gösterimler sonlu bir planlama çevreni içinde kaplama, akış korunumu, enerji tüketimi ve bütçe kısıtlarını dikkate almaktadırlar. Daha sonra, problemi kolayca çözebilmeyi sağlayan geçerli eşitsizlikler önerilmektedir. Problemin yapısı sebebiyle, dikkate değer bir süre içinde küçük örnekler için dahi en iyi çözümler hesaplanamamakta ve doğrusal programlama gevşetmeleri zayıf üst sınırlar vermektedir. Bu sebeple, Lagrange gevşetmesi ve açgözlü seçim ölçütü gibi teknikleri kullanan sezgisel yöntemler geliştirilmektedir. Yapılan bilgisayısal deneyler bu sezgisel yöntemlerin doğru ve etkin olduğunu göstermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

a_{ijk}	one if a type-k sensor located at point i can cover point j, zero otherwise
b_{ilj}	one if point j is within the communication range of a type-k sensor located at point i, zero otherwise
B	total available budget
c_{jk}	cost of placing a type-k sensor on point j
e_k^c	energy consumption of a type-k sensor for transmitting one unit of flow
e_k^r	energy consumption of a type-k sensor for receiving one unit of flow
e_k^s	energy consumption of a type-k sensor located at point i for sensing and processing during a period
E_k	initial battery energy of a type-k sensor
f_i	coverage quality requirement for point i
g_{jkt}	one if a type-k sensor located at point j is active in period t, zero otherwise
h_{jk}	number of data packets generated by a type-k sensor located at point j
K	number of sensor types
$\boldsymbol{K}$	index set for sensor types
$\boldsymbol{K'}$	index set for sensor and sink types
L	lifetime of the wireless sensor network
M_1	sufficiently large number
M_2	sufficiently large number
N	number of candidate locations
$\boldsymbol{N}$	index set for sensor and sink locations
n_t	one if period t is within the lifetime L, zero otherwise
r_k^c	communication range for a type-k sensor
r_k^s	sensing range for a type-k sensor
S	number of sinks

T	planning horizon
$\boldsymbol{T}$	index set for periods
u_{ijkt}	one if a type-k sensor located at point j is assigned to a sink located at point i in period t, zero otherwise
w_{iljkt}	one if there is a flow from a type-l sensor located at point i to a type-k sensor located at point j in period t, zero otherwise
x_{jk}	one if a type-k sensor is placed at point j, zero otherwise
y_{iljkt}	amount of flow from a type-l sensor located at point i to a type-k sensor located at point j in period t
z_{jkt}	one if a type-k sensor located at point j is active in period t, zero otherwise
δ_{viljkt}	Lagrangean dual variable for sink assignment constraints
ϵ_{jkt}	Lagrangean dual variable for copy constraints
λ_{it}	Lagrangean dual variable for coverage constraints
μ_{it}	Lagrangean dual variable for sink flow constraints
θ	Lagrangean dual variable for budget constraint
BIP	binary-integer programming
DH	discrimination heuristic
GH	greedy heuristic
LH_1	first Lagrangean heuristic
LH_2	second Lagrangean heuristic
LPSRPC	sink location, sensor placement, scheduling and routing problem under connectivity restrictions
LS	local search heuristic
MILP	mixed-integer linear program
PSRPC	sensor placement, scheduling and routing problem under connectivity restrictions
RP	routing problem
SPP	sensor placement problem
TS	tabu search heuristic
WSN	wireless sensor network

1. INTRODUCTION

A *sensor* is an inexpensive, low-power electronic device that can sense its neighborhood, process the information as data packets and communicate with the other sensors that are close enough. Depending on its type, a sensor may sense and process temperature, humidity, light, vibration, sound, radiation and many other factors (Karl and Willig, 2003). A *Wireless Sensor Network* (WSN) is a network in which sensors are deployed over the interested region, i.e. the *sensor field*, possibly remote or inaccessible to humans. WSNs are used currently for battlefield surveillance in military, forest fire detection in environmental sciences, monitoring of human physiological data in health (Akyıldız et al., 2002). A sensor collects information within its *sensing range*, after processing the data transmits to a base station namely a *sink* either directly or through other sensors that are within its *communication range*. The specifications related with the sensing and communication ranges, cost and the initial battery energy of the sensor can change depending on the type of the sensor.

The sensors in a WSN may be identical, in this case we have a *homogeneous* network. On the other hand, we can have different types of sensors in a *heterogeneous* network, which means their technical specifications and costs can be distinct. A sensor can be in either *active* or *standby* mode. In the active mode, sensor can perform sensing, processing and communicating activities. A standby sensor does not perform any of these activities but operates at the minimum energy level. A sensor consumes energy for sensing, information processing, data receiving and transmitting according to the technical characteristics of its type.

Each sensor in the network has limited battery energy and become obsolete when it is out of energy. Since budget is limited and a sensor consumes energy for collecting information, processing and transmitting the data, the network has a finite lifetime. In this thesis, we aim to maximize the lifetime of a WSN by determining the optimal locations of sensors and sinks, periods to be active for each sensor in order to use its energy economically, i.e. their activity schedules, assignments of sinks for each active

sensor and sensor-to-sink data flow routes.

The structure of the sensor field may be different from application to application. It can consists of discrete points, a two or three dimensional region as for the cameras in an art galery (Cardei and Wu, 2006). In this study, we assume that the sensor field is consisting of discrete points. Each point has to be covered by at least a certain number of sensors according to their importance level. For a critical point, it may be preferable to cover this point with numerous sensors to be secure in a sensor failure case. We have $uniform$ coverage if coverage quality requirements are the same for all points in the sensor field and $differentiated$ coverage if the coverage quality requirements vary.

One of the design issues of a WSN is to determine the locations of the active sensors in a period in order to satisfy the coverage quality requirements of each point in the sensor field. This is also known as *Sensor Placement Problem* (SPP) (Sahni and Xu, 2005). In our work, it is assumed that the coverage requirements of the points in the network can be different which gives arise to a $differentiated$ CP, otherwise we would have a $uniform$ CP. Depending on the characteristics of the sensor field, the locations of the sensors can be *random*, i.e. we cannot know the locations of the sensors a priori, or *deterministic* which means we can place a sensor exactly where we have decided. Our focus will be on deterministic sensor placement case. An effective sensor placement should use less budget as possible.

The limited battery energy of the sensors forces to consume energy economically for the sensor network to operate for a long time. Therefore, the deployed sensors have to remain in standby mode to save energy when they are not necessary in providing coverage of the points and keeping connectivity among the sensors. For each time period, the determination of the active and standby sensors among the deployed sensors to maximize the network lifetime while satisfying the coverage requirements of all points and staying communicated is the problem of activity scheduling of sensors (Wang et al., 2009). The schedule of the active and standby sensors in a period affects the network lifetime since scheduling is a way to use the limited energy efficiently.

Transmission of the collected information in the form of data packets from sensors to their assigned sinks also uses energy. The energy requirement for the transmission of a data packet generated by a sensor that is far from its assigned sink will be larger than the amount for a data packet generated by a sensor that is close to its sink. Hence, when we are given the locations of sinks and sensors with their assignments, we can determine the least energy consuming sensor-to-sink data flow routes. This problem is addressed as the *Routing Problem* (RP) (Schurgers and Srivastava, 2001).

Locations of the sinks are substantial for the lifetime of the network since they affect the sensor-to-sink data flow routes directly. A sink should be close to the sensors that are assigned to itself in order to minimize the amount of energy spent during the data transmission. This problem is referred to as the *Sink Location Problem* (SLP) (Liu and Xu, 2010).

In this thesis, we first concentrate on the *Sensor Placement, Scheduling and Routing Problem under Connectivity Restrictions* (PSRPC) which assumes that sink locations are given but determines the deployments and activity schedules of the sensors and establishes the sensor-to-sink data flow routes using the sink assignments decided for each sensor in order to maximize the network lifetime. We introduce a mixed-integer linear programming (MILP) model for the PSRPC. We provide a Lagrangean relaxation based heuristic solution procedure in order to solve large PSRPC instances. The heuristic gives a feasible solution which will be a lower bound and a relaxed solution which will be an upper bound on the optimal lifetime value of the PSRPC instance. Then, we add the deterrmination of optimal sink locations to the problem to obtain the *Sink Location, Sensor Placement, Scheduling and Routing Problem under Connectivity Restrictions* (LPSRPC) and propose a solution procedure based on a search on possible sink locations using a solution of PSRPC.

In the remainder of the thesis, we first briefly review the studies in the literature related with sensor placement, sensor activity scheduling, data routing and sink location in the next chapter. We provide MILP formulations for PSRPC and propose some valid inequalities for the main MILP formulation in the third chapter. In the fourth

chapter, we give the details of two different heuristic solution methods making use of the Lagrangean relaxation approach and subgradient algorithm for the PSRPC. In the fifth chapter, we extend our mathematical model to include the sink location problem, namely LPSRPC, and establish two different heuristic solution procedures to solve the LPSRPC instances. The sixth chapter reports the experimental results for proposed MILP models obtained with the introduced solution procedures. Finally the seventh chapter concludes the thesis.

2. LITERATURE SURVEY

2.1. Introduction

In this chapter, we present the previous studies on the design issues of wireless sensor networks. The works on locating sensors to cover each point in the sensor field by at least its required coverage quality are listed in Section 2.2, determining the activity schedules of each sensor during the lifetime of the network are summarized in Section 2.3, assigning at least one sink for each active sensor and finding the sensor-to-sink data flow routes are included in Section 2.4, locating the sinks in the sensor field are given in Section 2.5 and the studies that are combining at least two of these design issues are presented in Section 2.6.

2.2. Sensor Placement

In Chakrabarty et al. (2002) an integer linear program (ILP) is proposed to determine the optimum locations of sensors. The model minimizes the total deployment cost of sensors of two types on a grid network in order to cover all grid points that have uniform coverage quality requirement. They develop a divide and conquer approach for the solution procedure. The model does not include energy consumption constraints of sensors and routing of the data packets. Besides, the exact solution of the proposed ILP becomes computationally difficult for large instances.

In Dhillon and Chakrabarty (2003) heuristic procedures are developed to locate the sensors on a grid sensor network assuming sensors can imprecisely detect changes. They consider different objectives such as the minimization of total sensors in the network, maximizing average coverage of points and maximizing the coverage of points that are covered with the least number of sensors. However, the study does not present any method for the energy consumption and data routing issues.

In Lin and Chiu (2005) the determination of the sensor locations for the uniform

coverage problem under budget constraints is addressed. They aim to locate the sensors accurately even the Euclidean distance between two grid points becomes very small. In order to minimize of maximum distance error, they develop a simulated annealing based solution strategy which can find optimum solution for small sensor fields and can give a feasible placement of the sensors for large sensor fields. Again the energy limitation of the sensors is not considered.

In Altınel et al. (2008) differentiated point coverage problem is formulated as a binary integer programming (BIP) model. In the formulation, they consider a heterogeneous sensor network and differentiated coverage requirements for grid points in the sensor field. The objective is to minimize the total cost of sensor deployment. Moreover, the model is applicable for perfect, imperfect and uncertain sensing cases. For the solution procedure, they develop Lagrangean relaxation based heuristic and approximation algorithms that can give good solutions even for large instances in three hours. In Wang and Zhong (2006) a similar ILP is analyzed and an approximation algorithm that converts the optimal solution for the linear relaxation of ILP to an integer solution for the original ILP problem is developed. On the other hand, both of the studies ignore the energy consumption and data flow problems.

In Yuan et al. (2008) the direction of the research is the coverage problem for target detection. In these sonar like systems, targets emit signals and sensors detect target by measuring the energy of signals which decays with the distance. A group of sensor report the signal emissions in their sensing range to a central sensor called cluster head. In the case there is a target in the sensing range of the sensors in a cluster, cluster head decides whether there is really a target or it is only a false alarm by fusing the incoming information from the sensors in its cluster. The problem is a probabilistic coverage case, since a target may not be detected after the fusion of the information. Their aim is to determine the locations of sensors to minimize the false alarm rate in the network. They generate cluster based divide and conquer approach for the solution procedure which can give results for large instances in acceptable durations.

2.3. Sensor Activity Scheduling

In Nakamura et al. (2005) the problem of extending network lifetime by minimizing the energy consumption of the sensors is discussed. They present a dynamic mixed integer linear programming (MILP) model to assure the coverage of the target area and connectivity for each period to minimize the energy consumption in flat networks. Their test results for the model show that their model is succesful to prolong the network lifetime. There are two major drawbacks of the work: it assumes that the locations of the sensors that can cover the target region are given and it does not consider the energy consumption in data flows.

In Pazand and Datta (2006) extending the lifetime of sensor networks through efficient activity scheduling of sensors is addressed. The sensors of k types are randomly and uniformly spread over the sensor field. They prefer to develop a sensor scheduling method independent from the location information, since it is costly to obtain, using a graph theoretical approach. They introduce minimum dominating sets of sensors each of which ensures the coverage of the network. The minimum dominating sets are built via a heuristic. A target node, that should be covered by the sensors, sends a "Hello" message to its neighborhood and detect the sensors that can cover itself through the reply messages from sensors. At each period of the network, only one dominating set is active to cover the sensor field and the other sensors are in sleep mode. Simulations are conducted for the experimental results of their method upto 300 sensors distributed over a 2500 m^2 region. The drawback of the study is that it assumes the located sensors satisfy the coverage requirements of the points in the sensor field. Moreover, although the sensors in a dominating set are connected from the construction of the sets, the routing of the data to the sink nodes is not discussed in the paper.

In Yang and Cardei (2010) energy efficient sensor scheduling problem to maximize network lifetime is studied. The solution method makes use of the property that covering all of the sensor field is not necessary in all periods within the network lifetime. Instead the active sensors receive directions from the sink node about which part of the sensor field should they collect information. According to this information, which is

updated at each period, the decision of activating new sensors or turning off the active sensors is done. For the system to be operable, we want the network to be globally connected and the coverage of the network is provided. They define dominating set as a subset of sensors such that every sensor is either in the subset or can communicate with a sensor in the subset. Then, they introduce connected dominating sets (CDS), which can satisfy the current coverage requirements, as the backbone of the network. The message from the sink is propagated by the sensors in a CDS to all active and standby sensors in the network. As the message spread over the sensor field, sensors in the required region turn to active mode to satisfy coverage requirements of the points and provide global connectivity. They make simulations which reveal that the average number of active sensors in the sensor field drops and the network lifetime is improved compared with the method in which the meassages are directly sended from sinks to sensors. Despite the study decreases the energy consumption by efficiently scheduling the activities of the sensors, it neglects the transmission of the data and the respective energy consumption. Besides, the determation of the sensor locations is not a design issue of the paper.

In Yardibi and Karasan (2008) similar to the work by Yang and Cardei, 100% coverage of the sensor field is not required. Therefore, minimization of energy used in partial coverage through scheduling the sensor activities can improve the network lifetime. For this purpose, they develop distributed adaptive sleep scheduling algorithm (DASSA) which does not require the location information of the sensors. The sink node solves a simple ILP problem to determine the activity schedules of the sensors that are closer to itself since all network flows have to pass through these sensors. The selected active sensors search for the neighbor sensors which have sufficient remaining energy to be activated. The algorithm provides the coverage of the target region by satisfying the connectivity of the operating sensors. The computational experiments indicate that the performance of DASSA is superior than the centralized sleep scheduling algorithm (CSSA) which uses the location information of the sensors.

In Fei et al. (2010) densely deployed sensor networks are studied. They investigate the coverage aware sensor scheduling problem using genetic algorithms. Under

uniform coverage requirements, they aim to optimally schedule the already located sensors in different time slots to maximize the overall coverage. They evaluate the performance of their algorithms under various planning horizons through simulations and observe that the algorithm can maximize the overall coverage of the sensor field. The energy consumption for data transmission is disregarded.

In Lin et al. (2010) the sensors are scheduled for prolonging the network lifetime with an ant colony system based method. The algorithm first finds the maximum number of disjoint sets of sensors each of which can fulfill the sensing coverage and network connectivity requirements at the same time for a heterogeneous sensor network. Then the incremental solution meachanism builds disjoint connected cover sets on the basis of well designed construction graph. For further efficiency of the method a local search process is also developed. Experimental results show that the proposed method can find high quality solutions at a fast speed for WSNs with different characteristics.

2.4. Data Routing

In Ergen and Varaiya (2006) given the locations of the sensors that satisfies the coverage constraints the problem of locating the relay sensors to decrease the energy consumption during communication. Their objective is to minimize the total energy consumption so that the network is operable during the desired lifetime. They first formulate the problem as a nonlinear programming problem. For the solution of the problem, they develop an approximation algorithm based on restricting the locations where the relay nodes are allowed to a square lattice. Their algorithm approximates the original problem with performance ratio of two by trading complexity. The drawback of this study is that they assume only one information collection point.

In Byun et al. (2006) the lifetime of the sensor network is maximized through energy efficient coverage maintenance strategy. Their solution method makes use of the probabilistic approaches for power conservation. Each sensor calculates a probability with the information of geographical density of the sensors at each period. Depending on this probability, a sensor decides to which sensors should the data packets be trans-

mitted. The simulations indicate the method is efficient, however this study does not try to locate the sensors to provide coverage but to operate sensor network that covers the sensor field energy efficiently as long as possible.

In Cheng et al. (2008) given a covered sensor field with the located sensors the problem of locating least number of relay sensors that are not sensing but communicating to provide the global connectivity of the network. They model the problem by a network optimization problem namely Steiner Minimum Tree with Minimum number of Steiner Points and bounded edge length (SMT-MSP). They develop two approximation algorithms, whose performance ratios are 3 and 2.5 respectively, to solve the problem. The aim of the study is only to obtain a globally connected network without dealing with the problems of coverage of the sensor field or routing the data flows to an accumulation sensor.

In Hua and Yum (2008) an optimal routing and data aggregation scheme for wireless sensor networks is proposed. The objective is to maximize the network lifetime by jointly optimizing data aggregation and routing. They adopt a model to integrate data agregation with the underlying routing scheme and present a smoothing approximation function for the optimization problem. The necessary and sufficient conditions for achieving the optimality are derived and a distributed gradient algorithm is designed accordingly. They show that the proposed scheme can significantly reduce the data traffic and improve the network lifetime. The distributed algorithm can converge to the optimal value efficiently under all network configurations. The work is not extended for multiple sink nodes and for sensors with sleeping mode.

In Güney et al. (2010) the determination of sensor-to-sink data flow routes is discussed. They formulate MILP models with different objective functions such as minimization of total routing energy and minimization of total cost for commodity flows. They aim to find the best locations of the sinks and information flow paths between sensors and sinks when sensor locations are given. They test the solution efficiency of their formulations and for the most efficient formulation they develop heuristics and lower bounding approaches. However, the study does not consider to

efficiently locate the sensors to cover the sensor field.

2.5. Sink Location

In Oyman and Ersoy (2004) the limited battery resource of the sensors tried to be managed by deploying more than one sink node to maximize the network lifetime. If the sink nodes are closer to the sensors that are sending information to itself, then the corresponding energy consumption in transmission of the data will be lower. Utilizing this idea, the study intends to find the locations of the sinks for the network to be operable for at least given number of periods. Given the candidate locations for the sinks, they develop an algorithm that decomposes the network into smaller sub-networks and a reconstruction algorithm that is applied after the occurance of energy failures to improve the network lifetime. The experiments are conducted through simulations which show their approach is successful in prolonging the network lifetime.

In Yang (2006) the candidate locations of the sinks are explored by Genetic Algorithms. They work on a grid sensor field with given locations of homogeneous sensors. For a given network lifetime they aim to find the best locations of the sinks to minimize the energy consumed in transmitting the data. The drawback of the paper is it assumes the network lifetime is known a priori.

In Poe and Schmitt (2009) the sink placement problem for large size of sensor networks is handled. In order to minimize energy consumption and consequently extend the network lifetime, they propose a local search technique that does not require the location information of the sensors. Their self-organized sink placement (SOSP) strategy sets up a group of communicating sensors for each sink among its n-hop distance neighbors. SOSP exhibits a good performance by applying locally optimal sink placement in the experiments.

2.6. Integrated Works

In Yang et al. (2006) the k-(Connected) Coverage Set (k-CCS/k-CS) problems with the objective of minimizing total energy consumption while obtaining k coverage for reliability is adressed. They consider a sensor network consisting of a set of sensors deployed randomly. They propose one global solution for k-CS and two non-global algorithms. The first one is a linear programming algorithm that uses a cluster-based approach to select backbone sensors to form a set. The second uses the pruning algorithm based on 2-hop neighborhood information. They analyze the performance of their algorithms through theoretical analysis and simulations. However the developed algorithms are for uniform coverage and they assume connectivity of the sensors with each other but not connectivity of sensors with sinks.

In Liu and Liang (2008) the maximization of the network lifetime is addressed. They start from an already studied idea which first generates disjoint subsets each of which can cover all of the targets and works through activating only the sensors in one of these subsets each time. They extend this approach by allowing overlaps among these subsets. They partition the entire lifetime of a sensor into several equal intervals and accepting the sensor to be contained by several subsets which satisfy both target coverage and sensor connectivity. Initially they analyze the energy consumption of sensors in a Steiner tree rooted at the base station and spanning the sensors in a subset. Then they develop a heuristic algorithm, which takes into account the remaining energy of the sensors, for the target coverage problem. They conduct experiments by simulations to evaluate the performance of the proposed algorithm. The experimental results show that the network lifetime delivered by their algorithm is extended with the improvement of network connectivity.

In Li and Gao (2008) design of k-coverage schedules for wireless sensor networks to maximize the network lifetime is addressed. In order to ensure the quality of surveillance consuming as low energy as possible, they investigate the Sensor Scheduling for k-Coverage (SSC) problem which requires to efficiently schedule the sensors to satisfy k-coverage for the monitored area throughout the whole network lifetime. They pro-

pose two heuristic algorithms under different scenarios. They evaluate their presented algorithms through simulations. The drawback of the study is that it does not include the connectivity issue.

In Liu et al. (2008) the reliability of network communication is focused. They consider the problem of maintaining k-connectivity of sensor network at minimum energy level while keeping only a subset of sensors active to save energy. In their proposed scheme, each sensor is assumed to have multiple power levels and neighbor proximity, means exact location information is not adopted. Firstly the network partition is attained by power based clustering and next sensors are divided into equivalent classes according to the role of data forwarding to different adjacent clusters. Then Node Scheduling and Power Adjustment (NSPA) algorithm selects a subset of sensors with different power levels to construct the local minimum energy graph while maintaining network connectivity. If the number of intra-cluster sensors which have adjacent clusters exceeds a certain threshold, k-NSPA is employed to obtain k-connected topology. The simulation shows that their scheme can maintain a k-connected network energy efficiently. Again the connectivity is forced among sensors but not with a sink.

In Chaudhry et al. (2010) evolutionary methods are considered to find the best locations of the sensors to meet multiple objectives such as achieving maximum uniform coverage and maximum connectivity while minimizing the network energy cost. A flexible algorithm for sensor placement (FLEX) is presented that uses evolutionary computational approach to solve multiobjective sensor placement optimization problem when the number of sensor nodes is not fixed and the maximum number of nodes is not known a priori. They use Pareto dominance for Pareto-optimal layouts with respect to the objectives. The flexibility of the algorithm is illustrated by solving the sensor placement problem for diferent applications like facility surveillance, coverage with and without obstacles, preferential surveillance and forming a clustering hierarchy. The drawback of the study is that they assume that the communication is among sensors but not with a base station.

In Türkoğulları et al. (2010a) the design issues of sensor placement for the cover-

age of the sensor field, their activity schedules and determination of sensor-to-sink data flow routes for wireless sensor networks are addressed. They propose a MILP model that formulates these network design issues to maximize the network lifetime in heterogeneous sensor networks with the requirement of differentiated coverage. Their solution procedure consists of a heuristic which first finds connected sensor sets with minimum cost satisfying the coverage constraints and then determines optimal sensor-to-sink data routes with optimal flow quantities. The computational experiments performed on various test instances indicate that the heuristic is efficient and accurate to solve the problem. The same problem is again investigated in Türkoğulları et al. (2010b) with a different solution approach. They present a two-phase heuristic, which solves the linear programming relaxation by column generation in the first phase and in the second phase contructs a feasible solution for the original problem using the columns obtained in the first phase. Their computational experiments illustrate the quality of their solution method.

3. SENSOR PLACEMENT, SCHEDULING AND ROUTING PROBLEM WITH CONNECTIVITY RESTRICTIONS

3.1. Introduction

In this chapter, a mixed-integer linear programming (MILP) formulation for the Sensor Placement, Scheduling and Routing Problem with Connectivity Restrictions (PSRPC) is presented. PSRPC aims to determine the optimal sensor locations, to find the activity schedules of the sensors, to assign at least one sink for each sensor and to build the data flow routes from each sensor to the corresponding sink to maximize the network lifetime subject to coverage, flow balance, energy and budget constraints.

3.2. Mathematical Programming Formulations

We consider a sensor field consisting of N points indexed as $i \in \boldsymbol{N} = \{1, ..., N\}$. The locations of the sensors and sinks can be determined from the same index set $\boldsymbol{N}$. The sensors can be of K different types indexed as $k \in \boldsymbol{K} = \{1, ..., K\}$ with known sensing (r_k^s) and communication (r_k^c) ranges. We are assuming that a sink is a special type of sensor, namely type-0. Therefore, we define the index set $\boldsymbol{K'} = \boldsymbol{K} \bigcup \{0\}$ including the type index for sink. The unit cost (c_{jk}) of placing a type-k sensor at point j is known. There is a total available budget of B monetary units. In a period t indexed as $t \in \boldsymbol{T} = \{1, ..., T\}$ a sensor can be active or standby. We are considering sufficiently long planning horizon T to determine the maximum lifetime over it. Therefore, planning horizon T becomes an upperbound on the lifetime L of the network. A type-k sensor has an initial battery energy of E_k units which will be consumed by e_k^s units for sensing the network, e_k^r units for receiving and e_k^c units for transmitting the data in the active periods of the sensor. In a period, an active type-k sensor located at point j generates h_{jk} data packets, where data packet is a unit that measures data volume.

Technical characteristics of the existing commercial sensors, such as (xbow, 2009), reveal that the energy expenditure of a standby sensor is negligible. We also assume that, we can save energy if we keep a sensor in standby mode instead of activating it without performing any of sensing points in the sensor field, receiving and processing the data and transmitting the data packets to other sensors or a sink. This assumption works until the periods are not exceptionally short durations. Sinks have unlimited battery energy, they can receive data packets from the active sensors but cannot sense points in the sensor field or transmit the data packets to other active sensors or sinks.

In the sequel indices i and j represent the sensor and sink locations, indices k and l characterize the sensor types and index t shows the period. For convenience, a type-k sensor located at point j will be reprented by the pair (j, k).

3.2.1. Main Model

The MILP model we introduce in this section, locates the sensors, determine their activity schedules, assigns a sink for each active sensor and determines the sensor-to-sink data flow routes while maximizing the WSN's lifetime. Locations of the sensors are determined with decision variables x_{jk}. The location variable is one if a type-k sensor is placed at point j of the sensor field and zero otherwise. The determination of the periods that a sensor is active or standby gives us its activity schedule. The activity schedule variable z_{jkt} becomes one if a sensor (j, k) is active at period t and zero otherwise. In a period each active sensor should be assigned to a sink in order to send the collected information from the sensor field. The decision variables related with sink assignments of the sensors are u_{ijkt}. A sink assignment variable takes value one if an active sensor (j, k) is assigned to a sink located at point i in period t and zero otherwise. The decision variable n_t is one if a period t is within the lifetime L, and zero otherwise. The information collected from the sensor field by the active sensors should be sent to the corresponding sinks through the sensor-to-sink data flow routes. The flow variable y_{iljkt} represents the amount of data that is sent from sensor (i, l) to sensor (j, k) in period t. Lastly, the variable L represents the lifetime of the WSN, which we aim to maximize. In the model, locations and activity schedules and

sink assignments of sensors are of consideration. The assumptions related with sinks provide some information about the decision variables. Firstly, the locations of the sinks are konown a priori. Therefore, for $k = 0$ we are given for which j locations x_{jk} is equal to one. Since sinks have infinite battery energy, we can assume that sinks are active in all periods which means the variable z_{j0t} is equal to one for all periods t within the lifetime L if there is a sink at location j. For convenience, a sink $(j, 0)$ is assigned to itself as its sink assignment, i.e. u_{ij0t} is equal to one for all periods t within the lifetime L if $i = j$ and zero otherwise. Sinks are the data collection points of the sensor field, which means they are expected to have data inflows but no outflows. As a result, taking $l = 0$ if there is a sink $(i, 0)$, then the flow variable y_{i0jkt} is equal to zero for all sensors (j, k) in all periods t within the lifetime L.

We define a_{ijk} to represent the nonnegative coverage coefficients in the model. The value of this coefficient can be determined depending on the sensing characteristics of the sensors, i.e. perfect, imperfect and probabilistic sensing, in the sensor field. In perfect sensing, a sensor can cover all of the points that are in its sensing range. In this case a_{ijk} is set to one if point i is within the sensing range of the sensor (j, k) and zero otherwise. Sinks, being a special kind of sensors, cannot contribute to the coverage of a point in the sensor field. This means a_{ij0} is equal to zero for all $i, j \in \boldsymbol{N}$.

Imperfect sensing implies sensing intensity of a sensor decreases as the distance increases in its sensing range. Then, the coverage coefficients can be calculated as $a_{ijk} = \lambda_k/(d_{ij})^{\theta_k}$ where d_{ij} is the Euclidean distance between points i and j. λ_k and θ_k are technological parameters of a type-k sensor. In probabilistic sensing a sensor can cover a point in its sensing range with a probability. The probability of sensing a point i from a sensor (j, k) can be given as $p_{ijk} = e^{-\alpha_k d_{ij}}$. Note that the sensing probability decreases as the distance d_{ij} between the sensor and point increases. Exponential decay parameter α_k is a sensor specific parameter that shows how fast the probability decreases. For a type-k sensor high α_k value indicates low quality. In both cases, the coverage coefficients appear to be nonnegative real numbers in $[0, 1]$. The definitions of the coverage coefficients for imperfect and probabilistic sensing cases do not change the structure of the mathematical model, but may affect the values of the other model

parameters such as f_i, coverage quality requirement of point i. The mathematical model that we will introduce in this chapter assumes perfect sensing, but it is also applicable for imperfect and probabilistic sensing cases after making the adjustments in the coverage coefficients.

The aim of the sensor network is to cover the targeted points in the sensor field by at least the required number of sensors. For this purpose, f_i represents the coverage quality requirement of a point i in the sensor field. Their value depends on the sensing characteristics and the type of coverage. They are positive integers for perfect sensing case and become nonnegative rational numbers for imperfect and probabilistic sensing. They are all same in uniform coverage and may have distinct values for differentiated coverage. Similar to the coverage coefficients a_{ijk}, the communication coefficients b_{ilj} depend on the communication characteristics of the sensors. In perfect communication b_{ilj} value is one if a sensor at point j is within the communication range of sensor (i, l) and zero otherwise. According to our assumptions, sinks cannot transmit data to another sensor or sink. Then, if we have a sink $(i, 0)$, i.e. $l = 0$, b_{i0j} will be equal to zero for all points j. If the communication characteristic of the sensors is imperfect or probabilistic, then the communication coefficients should be updated appropriately as in the coverage coefficients. In our model, we assume perfect communication. Observe that the PSRPC model introduced below cannot be generalized to imperfect or probabilistic communication cases easily.

Table 3.1. Index sets used in the model

Index sets	Definition
$\boldsymbol{N}$	Index set for sensor and sink locations
$\boldsymbol{K}$	Index set for sensor types
$\boldsymbol{K'}$	Index set for sensor and sink types
$\boldsymbol{T}$	Index set for periods

There are different definitions used in literature for connectivity of a network. A network can be considered to be connected if the active sensors can communicate at least one of the other active sensors (Wang and Xiao, 2006). Another definition for

Table 3.2. Decision variables used in the model

Decision variables	Definition
L	Lifetime of the WSN
n_t	One if period t is within the lifetime L, zero otherwise
x_{jk}	One if a type-k sensor is placed at point j, zero otherwise
z_{jkt}	One if a sensor (j,k) is active in period t, zero otherwise
u_{ijkt}	One if a sensor (j,k) is assigned to a sink located at point i in period t, zero otherwise
y_{iljkt}	Amount of data flow from sensor (i,l) to sensor (j,k) in period t

connectivity of the network is the existance of a path from each active sensor to a sink node in order to transmit the data packets (Arai et al., 2010). In our study, a network is assumed to be connected in a period t if we can assign at least one sink for each active sensor and construct data flow paths from each sensor to its assigned sink. Table 3.1, Table 3.2 and Table 3.3 summarize the index sets, decision variables and model parameters used in our formulation, respectively.

Table 3.3. Parameters used in the model

Parameters	Definition
a_{ijk}	One if point i is within the coverage range of sensor (j,k), zero otherwise
b_{ilj}	One if a sensor located at point j is within the communication range of sensor (i,l), zero otherwise
B	Total available budget
c_{jk}	Cost of placing a type-k sensor at point j
e_k^c	Energy consumption of a type-k sensor for transmitting one unit of flow
e_k^r	Energy consumption of a type-k sensor for receiving one unit of flow
e_k^s	Energy consumption of a type-k sensor for sensing and processing during a period
E_k	Initial battery energy of a type-k sensor
f_i	Coverage quality requirement for point i
h_{jk}	Number of data packets generated by sensor (j,k) per period
N	Number of candidate locations for sensors and sinks
K	Number of sensor types
r_k^c	Communication range of a type-k sensor
r_k^s	Sensing range of a type-k sensor
T	Planning horizon

The problem of placing sensors, scheduling and routing with communication restrictions (PSRPC) under coverage, energy and budget constraints in order to maximize the network lifetime can be modeled as the following MILP:

$PSRPC$:

$$\max \quad L \tag{3.1}$$

s.t.

$$L \geq 1 \tag{3.2}$$

$$Tn_t \geq L + 1 - t \qquad t \in \boldsymbol{T} \tag{3.3}$$

$$\sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} a_{ijk} z_{jkt} \geq f_i n_t \qquad i \in \boldsymbol{N};\ t \in \boldsymbol{T} \tag{3.4}$$

$$\sum_{i \in \boldsymbol{N}} \sum_{l \in \boldsymbol{K}} y_{iljkt} + h_{jk} z_{jkt} = \sum_{i \in \boldsymbol{N}} \sum_{l \in \boldsymbol{K'}} y_{jkilt} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K};\ t \in \boldsymbol{T} \tag{3.5}$$

$$\sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} y_{jki0t} = \sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} h_{jk} u_{ijkt} \qquad i \in \boldsymbol{N};\ t \in \boldsymbol{T} \tag{3.6}$$

$$\sum_{t \in \boldsymbol{T}} (e_k^s z_{jkt} + e_k^r \sum_{i \in \boldsymbol{N}} \sum_{l \in \boldsymbol{K}} y_{iljkt} + e_k^c \sum_{i \in \boldsymbol{N}} \sum_{l \in \boldsymbol{K'}} y_{jkilt}) \leq E_k \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K} \tag{3.7}$$

$$\sum_{i \in \boldsymbol{N}} \sum_{l \in \boldsymbol{K'}} y_{jkilt} \leq M_1 z_{jkt} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K};\ t \in \boldsymbol{T} \tag{3.8}$$

$$\sum_{i \in \boldsymbol{N}} \sum_{l \in \boldsymbol{K}} y_{ilj0t} \leq M_2 x_{j0} \qquad j \in \boldsymbol{N};\ t \in \boldsymbol{T} \tag{3.9}$$

$$z_{jkt} \leq x_{jk} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K};\ t \in \boldsymbol{T} \tag{3.10}$$

$$z_{jkt} \leq n_t \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K};\ t \in \boldsymbol{T} \tag{3.11}$$

$$u_{ijkt} \leq x_{i0} \qquad i, j \in \boldsymbol{N};\ k \in \boldsymbol{K};\ t \in \boldsymbol{T} \tag{3.12}$$

$$u_{ijkt} \leq z_{jkt} \qquad i, j \in \boldsymbol{N};\ k \in \boldsymbol{K};\ t \in \boldsymbol{T} \tag{3.13}$$

$$\sum_{i \in \boldsymbol{N}} u_{ijkt} \geq z_{jkt} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K};\ t \in \boldsymbol{T} \tag{3.14}$$

$$\sum_{i \in \boldsymbol{N}} u_{ijkt} \leq 1 \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K};\ t \in \boldsymbol{T} \tag{3.15}$$

$$u_{vilt} \leq u_{vjkt} \; if \; y_{iljkt} > 0 \qquad v, i, j \in \boldsymbol{N};\; k \in \boldsymbol{K'}; l \in \boldsymbol{K}; t \in \boldsymbol{T} \quad (3.16)$$

$$y_{iljkt} \leq M_2 b_{ilj} \qquad i, j \in \boldsymbol{N};\; k \in \boldsymbol{K'};\; l \in \boldsymbol{K};\; t \in \boldsymbol{T} \quad (3.17)$$

$$\sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} c_{jk} x_{jk} \leq B \quad (3.18)$$

$$n_t, x_{jk}, z_{jkt}, u_{ijkt} \in \{0, 1\}, L \geq 0, y_{iljkt} \geq 0. \quad (3.19)$$

Objective function (3.1) maximizes the network lifetime. Constraint (3.2) eliminates the trivial but nonsense solution $\{\mathbf{n = 0,\ x = 0,\ z = 0,\ u = 0,\ y = 0}\}$, which gives $L = 0$, from the solution space. The solution is not meaningful since a realistic WSN is expected to operate at least one period. Constraints (3.3) determine the periods that are within the lifetime L and constraints (3.4) try to activate the sensors to satisfy the coverage quality requirement f_i of each point i in the sensor field for these periods. One important observation is when the time is beyond the lifetime, i.e. $t > L$, constraints (3.3) set n_t to 0 which makes coverage constraints (3.4) redundant. Constraints (3.5) ensure the data flow balance for each sensor (j, k) in each period t. If a sensor (j, k) is active in period t, then it adds h_{jk} units of flow, i.e. data packets, to the incoming data flow from the active sensors and sends the total flow to other active sensors or a sink as outflow. Constraints (3.6) consider the incoming data flow to a sink located at point i. Since we assign a sink for each sensor (j, k) and each active sensor initiates h_{jk} units of data flow, we expect that the data flow of each active sensor will be reached to the corresponding sink. This means each active sensor contributes by h_{jk} units to the inflow of its assigned sink which generates the constraints (3.6). In a period an active sensor consumes energy for sensing and processing the data collected from the sensor field, for receiving data from other active sensors and transmitting data to other active sensors or a sink. Total consumed energy of a type-k sensor during its active periods is limited by the initial battery energy E_k, which is modeled with the energy constraints (3.7). Constraints (3.8) guarantee that there is no outflow from a sensor (j, k) in standby mode. Besides, total outflow from an active sensor in a period is bounded by M_1, where M_1 is a sufficiently large number. We define M_1 for this model as $(\max_{j,k} h_{jk})N(K + 1)$, since there are N different candidate locations and $K + 1$ different sensor types including the sink type to place an active sensor, to which sensor

(j,k) can transmit at most $(\max_{j,k} h_{jk})$ units of data packets in a period. Another bound on flow variables is given by the constraints (3.9). In order to send a flow to a sink $(j,0)$, there should be a sink at point j. Total inflow to an existing sink can be at most M_2 which is determined as $(\max_{j,k} h_{jk})NK$ for our model. The reason for the difference among M_1 and M_2 is that there cannot be outflow from a sink. Constraints (3.10) force to deploy a sensor before activating it and constraints (3.11) guarantee that a sensor is not active for the periods out of the lifetime L. A feasible sink assignment of a sensor (j,k) to a sink $(i,0)$ in a period t requires an active sensor (j,k) and a sink located at point i which can be provided by the constraints (3.12) and (3.13). Moreover, according to constraints (3.14) and (3.15) we can find one and only one sink for each active sensor in a period t. One may argue why there is unique sink for each active sensor in a period t. We prefer to model for a unique sink since it will simplify the determination of the flow routes from sensors to sinks in the solution procedure. The active sensors can send their data to the assigned sink through the other active sensors. Then, a sensor (i,l) can have a sink assignment the same with one of the sensors to which sensor (i,l) sends flow. Constraints (3.16) make use of this idea and bound the sink assignment variables of sensor (i,l), i.e. u_{vilt}, with the ones of sensor (j,k), i.e. u_{vjkt}, if there is a flow from sensor (i,l) to sensor (j,k). Communication range of a sensor (i,l) determines the candidate sensors to which it can send flow in a period. Constraints (3.17) give a bound to the flow from sensor (i,l) to sensor (j,k) with the corresponding communication coefficient b_{ilj}. Constraints (3.18) force that the total deployment cost does not exceed the total available budget. Finally, constraints (3.19) put the nonnegativity restriction on the decision variables. In the model, constraints to limit the number of sensors located at the same point are not included since we observe that in the optimal solution two or more sensors are rarely active at the same point within the same period while it is not an energy efficient activity schedule.

Note that the above formulation is not linear because of the constraints (3.16). On the other hand, the formulation can be linearized by introducing binary variables $w_{iljkt} \in \{0,1\}$ as follows:

$$y_{iljkt} \leq M_2 w_{iljkt} \qquad i,j \in \boldsymbol{N};\ k \in \boldsymbol{K'};\ l \in \boldsymbol{K};\ t \in \boldsymbol{T} \quad (3.20)$$

$$u_{vilt} - u_{vjkt} \leq M_2(1 - w_{iljkt}) \qquad v,i,j \in \boldsymbol{N};\ k \in \boldsymbol{K'};\ l \in \boldsymbol{K};\ t \in \boldsymbol{T} \quad (3.21)$$

We can still make some modifications with the above constraints (3.21) through the sink assignment variables u_{ij0t}. As it is discussed above, the sink assignment variables for sinks are not actually decision variables since we know their values when we are given the locations of sinks. Therefore, we can simply drop u_{ij0t} variables and propose the following set of constraints instead of (3.21):

$$u_{vilt} - u_{vjkt} \leq M_2(1 - w_{iljkt}) \qquad v,i,j \in \boldsymbol{N};\ k,l \in \boldsymbol{K};\ t \in \boldsymbol{T} \quad (3.22)$$

$$u_{vilt} - \mathbf{1}_j(v) x_{j0} \leq M_2(1 - w_{ilj0t}) \qquad v,i,j \in \boldsymbol{N};\ l \in \boldsymbol{K};\ t \in \boldsymbol{T} \quad (3.23)$$

The characteristic function $\mathbf{1}_j(v)$ is defined as:

$$\mathbf{1}_j(v) = \begin{cases} 1, & \text{if } v = j; \\ 0, & \text{otherwise.} \end{cases} \quad (3.24)$$

The replacement strategy makes use of our previous assumption that u_{ij0t} is equal to one if $i = j$ and zero otherwise. When there is a flow from an active sensor (i,l) to a sink $(j,0)$, we may assign sink $(j,0)$ to this sensor, i.e. $u_{jilt} \leq 1$, but it does not mean that we can assign this sensor to another sink, i.e. $u_{vilt} \leq 0$ for $v \neq j$.

In order to visualize the problem better we can make some observations related with the mathematical formulation. Let $inflow_{jk} = \sum_{i \in \boldsymbol{N}} \sum_{l \in \boldsymbol{K}} y_{iljkt}$ and $outflow_{jk} = \sum_{i \in \boldsymbol{N}} \sum_{l \in \boldsymbol{K'}} y_{jkilt}$. Notice that, if we decide to locate a sensor (j,k), i.e. $z_{jkt} = 1$, in period t then it is necessary to assign a sink to it by constraints (3.14) and there will be some outflow from the sensor, i.e. $outflow_{jk} > 0$, by constraints (3.5). On the

other hand, if a sensor (j,k) is not active, i.e. $z_{jkt} = 0$, in period t then there cannot be outflow from the sensor, i.e. $outflow_{jk} = 0$, by constraints (3.8). Besides, the flow balance constraint (3.5) for sensor (j,k) implies $inflow_{jk} = outflow_{jk}$ from which we conclude $inflow_{jk} = 0$ also. This means, until a sensor (j,k) is not active in period t, there will not be any incoming and outgoing flows for sensor (j,k).

Proposition 3.1. *If there is a flow from sensor (j,k) to sink $(i,0)$ in period t, then there cannot be any outflow from sensor (j,k) to another sink directly and sink $(i,0)$ is the unique sink assignment for sensor (j,k) in period t, i.e. $u_{ijkt} = 1$.*

Proof. Let $(i_1,0)$ and $(i_2,0)$ be two different sinks in the sensor field. Assume for contradiction that there are positive outflows from sensor (j,k) to both of the sinks directly, i.e. $y_{jki_10t} > 0$ and $y_{jki_20t} > 0$. We have defined that a sink is assigned to itself as its sink assignment, meaning $u_{i_1i_10t} = 1$ and $u_{ji_10t} = 0$ for all $j \neq i_1$ and similarly for sink $(i_2,0)$. Since $y_{jki_10t} > 0$, constraints (3.16) imply $u_{vjkt} \leq u_{vi_10t}$ for all v and similarly $u_{vjkt} \leq u_{vi_20t}$ for all v. Assume without loss of generality that sensor (j,k) is assigned to sink $(i_1,0)$ in period t, i.e. $u_{i_1jkt} = 1$. Besides, from constraints (3.15) $u_{vjkt} = 0$ for all $v \neq i_1$. However, for $v = i_1$ the inequality $u_{i_1jkt}(=1) \leq u_{i_1i_20t}(=0)$ will give a contradiction. Hence, an active sensor (j,k) can send flow directly to only one sink which is its unique sink assignment. □

Proposition 3.2. *If there is a flow from sensor (j_1,k_1) to sink $(i_1,0)$ in period t, then there cannot be any outflow from sensor (j_1,k_1) to another sensor (j_2,k_2) that has sink assignment different than the one of sensor (j_1,k_1) in period t.*

Proof. Let $(i_1,0)$ and $(i_2,0)$ be two different sinks in the sensor field. From Proposition (3.1) sink $(i_1,0)$ is the unique sink assignment of sensor (j_1,k_1) in period t, i.e. $u_{i_1j_1k_1t} = 1$ and $u_{vj_1k_1t} = 0$ for all $v \neq i_1$. Assume sink $(i_2,0)$, different than sink $(i_1,0)$, is the unique sink assignment of sensor (j_2,k_2) in period t, i.e. $u_{i_2j_2k_2t} = 1$ and $u_{vj_2k_2t} = 0$ for all $v \neq i_2$. If there is a flow from sensor (j_1,k_1) to sensor (j_2,k_2) in period t, i.e. $y_{j_1k_1j_2k_2t} > 0$, then from constraints (3.16) we should have $u_{vj_1k_1t} \leq u_{vj_2k_2t}$ for all v. However, taking $v = i_1$ gives $u_{i_1j_1k_1t}(=1) \leq u_{i_1j_2k_2t}(=0)$, which is a contradiciton. Hence the results follows. □

Proposition 3.3. *Assuming a sensor (j,k) is assigned to a sink $(v,0)$, there cannot be any outflow from sensor (j,k) to a sensor (i,l), which is assigned to another sink in period t.*

Proof. Assume sensor (j,k) is assigned to sink $(v_1,0)$, i.e. $u_{v_1jkt}=1$, and sensor (j,k) is assigned to sink $(v_2,0)$, i.e. $u_{v_2ilt}=1$, in period t. By constraints (3.14) these are unique sink assignments of sensors (j,k) and (i,l), respectively. Assume for contradiction there is a flow from sensor (j,k) to sensor (i,l), $y_{jkilt}>0$. Then, from constraints (3.16) we have $u_{vjkt}\leq u_{vilt}$ for all v. Taking $v=v_1$ gives $u_{v_1jkt}(=1)\leq u_{v_1ilt}(=0)$ which is a contradiction. □

Notice that the above formulation cannot eliminate flow loops in the network. This means an active sensor (i,l) sends flow to another active sensor (j,k), then sensor (j,k) sends this flow back to the sensor (i,l) and so on. This kind of repetition in flow of data consumes the battery energy without a valuable output. Moreover, consider an active sensor (i,l) and two other active sensors, say sensors (j,k) and (r,m), in the communication range of sensor (i,l). In such a case sensor (i,l) can send data flow to one or both of the sensors (j,k) and (r,m) in that period. From the point of view of sensor (i,l), energy consumption for transmitting a data packet will be the same for all alternatives of sending its data packets, since both sensors are in the communication range of sensor (i,l). Therefore, without loss of generality we can assume that there can be only one outflow path for an active sensor in a period. This means an active sensor (i,l) will choose only one active sensor in its communication range, say sensor (j,k) with $b_{ilj}=1$, to send its flow. The following set of constraints aim to eliminate flow loops by forcing a unique outflow path from a sensor:

$$\sum_{j\in \boldsymbol{N}}\sum_{k\in \boldsymbol{K'}} w_{iljkt}\leq 1 \qquad\qquad i\in \boldsymbol{N};\ l\in \boldsymbol{K};\ t\in \boldsymbol{T} \quad (3.25)$$

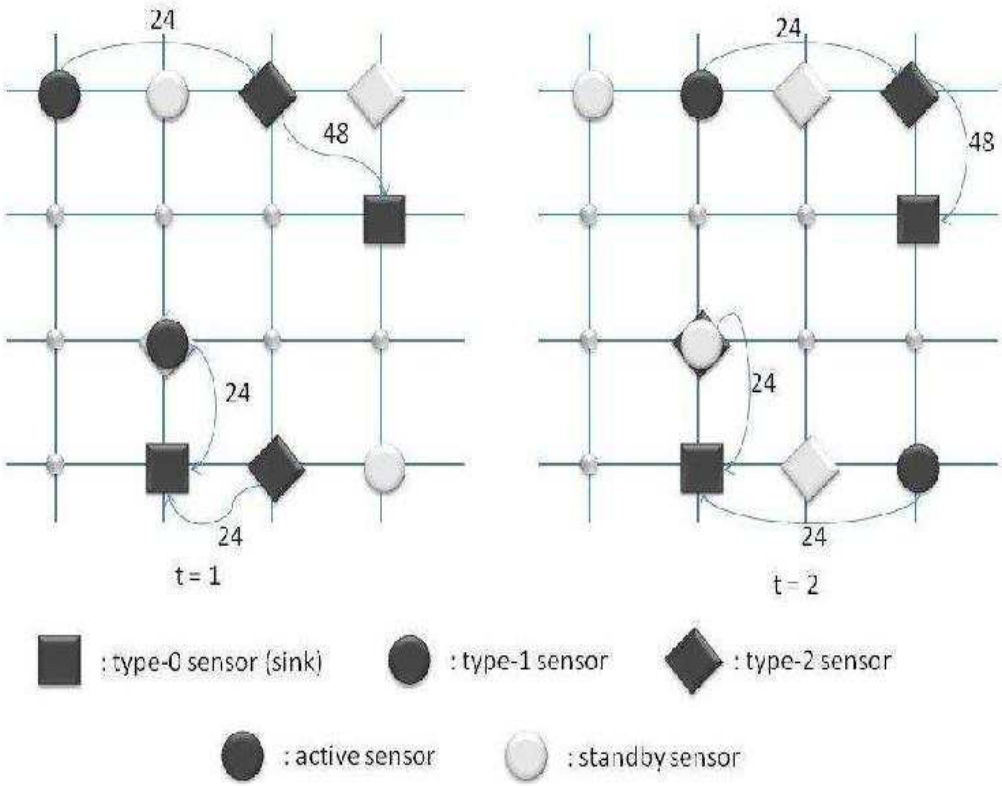

Figure 3.1. A sample sensor network operating for T = 2 periods

In order to illustrate the problem, we can consider a sample solution for a problem instance defined on a 4 × 4 grid whose model parameters can be given as: we have two different sensor types (i.e. $K = 2$) and two sinks, planning horizon is equal to two (i.e. $T = 2$), coverage quality requirement is one for each point (i.e. $f_i = 1$ for all $i \in \boldsymbol{N}$), sensing range of type-1 sensors is one and half of the sensing range of type-2 sensors (i.e. $2r_1^s = r_2^s = 2$), communication range of type-1 sensors is two and half of the communication range of type-2 sensors (i.e. $2r_1^c = r_2^c = 4$) and each active sensor can transmit $h_{jk} = 24$ data packets in a period.

Notice that in Figure 3.1, activity schedules of the deployed sensors may change from one period to another. Active sensors collect data from the points in their sensing range, process them as data packets and transmit the data either directly or through other active sensors to the corresponding sink. There can be only one outflow from a

sensor which prevents loops in the flow routes. Sinks have unlimited energy capacity and are active in all periods without contributing to the sensing process.

3.2.2. Formulations with Alternative Objectives

The formulation given in the previous subsection tries to maximize the lifetime of the WSN. It is also possible to give mathematical formulations which use other objective functions such as the minimization of total deployement cost and minimization of total energy consumption. For the sake of completeness, the MILP formulations for these two objectives will be given here, but solution procedure will not be analyzed in this study.

The following MILP formulation aims to minimize the total deployment cost.

$PSRPC_1$:

$$\min \quad \sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} c_{jk} x_{jk} \tag{3.26}$$

s.t. (3.2) - (3.15),(3.17), (3.19), (3.20), (3.22), (3.23), (3.25).

The other MILP formulation tries to minimize the total energy consumption over all periods.

$PSRPC_2$:

$$\min \quad \sum_{t \in \boldsymbol{T}} (e_k^s z_{jkt} + e_k^r \sum_{i \in \boldsymbol{N}} \sum_{l \in \boldsymbol{K}} y_{iljkt} + e_k^c \sum_{i \in \boldsymbol{N}} \sum_{l \in \boldsymbol{K'}} y_{jkilt}) \tag{3.27}$$

s.t. (3.2) - (3.6), (3.8) - (3.15), (3.17) - (3.19), (3.20), (3.22), (3.23), (3.25).

3.3. Valid Inequalities

The MILP formulation of PSRPC proposed in the previous subsection is sufficient to define the feasible solution space. In this section, we give some other set of constraints which are also valid for the convex hull of the mixed integer set, which we will call as Ω. These valid inequalities are not necessary for the problem formulation but they will hopefully improve the performance of the solution procedure. The convex hull of the mixed integer set, namely Ω, is generated by the constraints (3.2) - (3.19), (3.20), (3.22), (3.23), (3.25) and defined as:

$$\Omega = \{L, \mathbf{n}, \mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{y}, \mathbf{w} : L, \mathbf{n}, \mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{y}, \mathbf{w}$$
$$\text{satisfy constraints } (3.2) - (3.15), (3.17) - (3.19), (3.20), (3.22), (3.23), (3.25)\}$$

Proposition 3.4.

$$\sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K'}} y_{iljkt} \leq \sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} h_{jk} z_{jkt} \qquad i \in \boldsymbol{N};\ l \in \boldsymbol{K};\ t \in \boldsymbol{T} \quad (3.28)$$

are valid inequalities for the convex hull Ω*.*

Proof. Intuitively, if a sensor (j, k) is not active in period t, there cannot be any outflow from this sensor in that period and if there is an outflow from an active sensor (j, k), it is at most the total flow contribution of active sensors. For a formal proof consider a feasible solution $(L, \mathbf{n}, \mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{y}, \mathbf{w})$ in Ω. Let *totalflow(t)* represents the total amount of inflow to all sinks in period t, i.e. $\sum_{i \in \boldsymbol{N}} \sum_{l \in \boldsymbol{K}} \sum_{j \in \boldsymbol{N}} y_{ilj0t}$. Obviously, *totalflow(t)* is the maximum possible outflow from a sensor (i, l). Then

$$\begin{aligned} \sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K'}} y_{iljkt} &\leq \sum_{i \in \boldsymbol{N}} \sum_{l \in \boldsymbol{K}} \sum_{j \in \boldsymbol{N}} y_{ilj0t} \\ &= \sum_{j \in \boldsymbol{N}} \sum_{i \in \boldsymbol{N}} \sum_{l \in \boldsymbol{K}} h_{il} u_{jilt} \quad \text{(by constraints (3.6))} \end{aligned}$$

We know from constraints (3.13) - (3.15) for a sensor (i, l) there is only one sink

assignment, say sink $(j_{il}, 0)$, in a period t. Hence

$$\sum_{j\in\boldsymbol{N}}\sum_{i\in\boldsymbol{N}}\sum_{l\in\boldsymbol{K}} h_{il}u_{jilt} = \sum_{i\in\boldsymbol{N}}\sum_{l\in\boldsymbol{K}} h_{il}u_{j_{il}ilt}$$
$$\leq \sum_{i\in\boldsymbol{N}}\sum_{l\in\boldsymbol{K}} h_{il}z_{ilt} \quad \text{(by constraints (3.13))}$$

which shows that constraints (3.28) are valid inequalities for Ω. □

Proposition 3.5.

$$w_{iljkt} \leq b_{ilj} \qquad i, j \in \boldsymbol{N};\ k \in \boldsymbol{K'};\ l \in \boldsymbol{K};\ t \in \boldsymbol{T} \quad (3.29)$$

are valid inequalities for the convex hull Ω*.*

Proof. We know from constraints (3.17) that an active sensor (i, l) can send flow to an active sensor (j, k) if it is in the communication range of sensor (i, l). In addition, from constraints (3.20) variable w_{iljkt} is forced to be one if flow variable y_{iljkt} is positive. For a formal proof consider a feasible solution $(L, \mathbf{n}, \mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{y}, \mathbf{w})$ in Ω. If $b_{ilj} = 1$ then flow variables $y_{iljkt} \geq 0$ by constraints (3.17) and $w_{iljkt} \in \{0, 1\}$ by constraints (3.20) which will be feasible with respect to constraints (3.29). If $b_{ilj} = 0$ then flow variables $y_{iljkt} = 0$ by constraints (3.17) and $w_{iljkt} \in \{0, 1\}$ by constraints (3.20). Setting $w_{iljkt} = 0$ is feasible with respect to constraints (3.29) and Ω. □

There are some other valid inequalities proposed in the literature that are also relevant for our problem (Türkoğulları, 2010c). We state them without proof in the following.

Proposition 3.6.

$$L - \sum_{t \in \boldsymbol{T}} n_t = 0 \tag{3.30}$$

$$n_{t-1} \geq n_t \qquad t \in \boldsymbol{T} \tag{3.31}$$

$$\sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} E_{jk} x_{jk} \geq E_{lb} L \tag{3.32}$$

$$\sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} \sum_{t \in \boldsymbol{T}} (e_k^s + h_{jk} e_k^c) z_{jkt} \geq E_{lb} L \tag{3.33}$$

$$\sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} c_{jk} z_{jkt} \geq C_{lb} n_t \qquad t \in \boldsymbol{T} \tag{3.34}$$

$$\sum_{t \in \boldsymbol{T}} z_{jkt} \leq \left\lfloor \frac{E_k}{(e_k^s + h_{jk} e_k^c) z_{jkt}} \right\rfloor x_{jk} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K} \tag{3.35}$$

are valid inequalities for the convex hull Ω, where E_{lb} denotes a lower bound on the optimal value of the following binary integer programming (BIP) problem:

$$\min \sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} \sum_{t \in \boldsymbol{T}} (e_k^s + h_{jk} e_k^c) p_{jk} \tag{3.36}$$

s.t.

$$\sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} a_{ijk} p_{jk} \geq f_i \qquad i \in \boldsymbol{N} \tag{3.37}$$

$$p_{jk} \in \{0, 1\} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K} \tag{3.38}$$

and C_{lb} is a lower bound on the optimal value of the following BIP problem:

$$\min \sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} \sum_{t \in \boldsymbol{T}} c_{jk} p_{jk} \tag{3.39}$$

s.t.

$$\sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} a_{ijk} p_{jk} \geq f_i \qquad i \in \boldsymbol{N} \tag{3.40}$$

$$p_{jk} \in \{0, 1\} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K} \tag{3.41}$$

Proof. See (Türkoğulları, 2010c). □

4. SOLVING THE SENSOR PLACEMENT, SCHEDULING AND ROUTING PROBLEM WITH CONNECTIVITY RESTRICTIONS

4.1. Introduction

The computational results related with exact solution procedure reported in Chapter 6 show that it is not efficient even for small instances of PSRPC. Therefore, heuristic solution techniques can be utilized. In this chapter two different Lagrangean relaxation strategy and the corresponding Lagrangean heuristics for PSRPC are introduced. We try to relax some of the complicating constraints using Lagrange multipliers and solve the Lagrangean dual problem using the subgradient algorithm. The objective function value of a Lagrangean subproblem will be an upper bound for the main problem since we are relaxing some of the constraints which give a larger solution space to maximize the lifetime. However, solutions obtained from the Lagrangean subproblems may not be feasible with respect to the relaxed constraints. Then, we can construct feasible solutions of PSRPC with some heuristic approaches to the solutions of the Lagrangean subproblems and reach a lower bound for objective function value of the main problem. Since we cannot find optimum solutions of PSRPC instances, our aim in this chapter is to develop Lagrangean heuristics that find tight lower and upper bounds for the PSRPC.

4.2. First Lagrangean Heuristic

We first add dummy constraints in order to replicate activity scheduling variables z_{jkt} with new variables $g_{jkt} \in \{0, 1\}$ to the MILP formulation introduced in section (3.2) which can be given as:

$$g_{jkt} \leq z_{jkt} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K};\ t \in \boldsymbol{T} \quad (4.1)$$

$$g_{jkt} \geq z_{jkt} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K};\ t \in \boldsymbol{T} \quad (4.2)$$

Then, we replace z_{jkt} variables in constraints (3.5), (3.7) and (3.8) with g_{jkt} variables. We aim to decompose the Lagrangean subproblem into smaller subproblems with this equivalent formulation.

The Lagrangean heuristic is based on the relaxation of the constraints (3.4), (3.6), (3.22), (3.23), (3.18), (4.1) and (4.2). Note that the constraints (3.6) are equalities. Therefore, in order to update the Lagrange multipliers in the sugradient algorithm together, we split these equality constraints into two as follows:

$$\sum_{j\in \boldsymbol{N}}\sum_{k\in \boldsymbol{K}} y_{jki0t} \leq \sum_{j\in \boldsymbol{N}}\sum_{k\in \boldsymbol{K}} h_{jk}u_{ijkt} \qquad i\in \boldsymbol{N};\ t\in \boldsymbol{T} \quad (4.3)$$

$$\sum_{j\in \boldsymbol{N}}\sum_{k\in \boldsymbol{K}} y_{jki0t} \geq \sum_{j\in \boldsymbol{N}}\sum_{k\in \boldsymbol{K}} h_{jk}u_{ijkt} \qquad i\in \boldsymbol{N};\ t\in \boldsymbol{T} \quad (4.4)$$

As a result, we relax the constraints (3.4), (4.3), (4.4), (3.22), (3.23), (3.18), (4.1) and (4.2) with multipliers $\boldsymbol{\lambda}\geq 0,\ \boldsymbol{\mu}^1\geq 0,\ \boldsymbol{\mu}^2\geq 0,\ \boldsymbol{\delta}^1\geq 0,\ \boldsymbol{\delta}^2\geq 0,\ \theta\geq 0,\ \boldsymbol{\epsilon}^1\geq 0$ and $\boldsymbol{\epsilon}^2\geq 0$, respectively, in order to obtain the Lagrangean subproblem:

$$\begin{aligned}
L_{UB}(\boldsymbol{\lambda},\boldsymbol{\mu}^1,\boldsymbol{\mu}^2,\boldsymbol{\delta}^1,\boldsymbol{\delta}^2,\theta,\boldsymbol{\epsilon}^1,\boldsymbol{\epsilon}^2) = \max L + \ & \theta\left(B-\sum_{j\in \boldsymbol{N}}\sum_{k\in \boldsymbol{K}} c_{jk}x_{jk}\right)\\
&+\sum_{i\in \boldsymbol{N}}\sum_{t\in \boldsymbol{T}}\mu^1_{it}\left\{\sum_{j\in \boldsymbol{N}}\sum_{k\in \boldsymbol{K}}(h_{jk}u_{ijkt}-y_{jki0t})\right\}\\
&+\sum_{i\in \boldsymbol{N}}\sum_{t\in \boldsymbol{T}}\mu^2_{it}\left\{\sum_{j\in \boldsymbol{N}}\sum_{k\in \boldsymbol{K}}(y_{jki0t}-h_{jk}u_{ijkt})\right\}\\
&+\sum_{v\in \boldsymbol{N}}\sum_{i\in \boldsymbol{N}}\sum_{l\in \boldsymbol{K}}\sum_{j\in \boldsymbol{N}}\sum_{k\in \boldsymbol{K}}\sum_{t\in \boldsymbol{T}}\delta^1_{viljkt}\left(M_2(1-w_{iljkt})-u_{vilt}+u_{vjkt}\right)\\
&+\sum_{v\in \boldsymbol{N}}\sum_{i\in \boldsymbol{N}}\sum_{l\in \boldsymbol{K}}\sum_{j\in \boldsymbol{N}}\sum_{t\in \boldsymbol{T}}\delta^2_{vilj0t}\left(M_2(1-w_{ilj0t})-u_{vilt}+\mathbf{1}_j(v)x_{j0}\right)\\
&+\sum_{i\in \boldsymbol{N}}\sum_{t\in \boldsymbol{T}}\lambda_{it}\left\{\left(\sum_{j\in \boldsymbol{N}}\sum_{k\in \boldsymbol{K}}a_{ijk}z_{jkt}\right)-f_in_t\right\}\\
&+\sum_{j\in \boldsymbol{N}}\sum_{k\in \boldsymbol{K}}\sum_{t\in \boldsymbol{T}}\epsilon^1_{jkt}(z_{jkt}-g_{jkt})+\sum_{j\in \boldsymbol{N}}\sum_{k\in \boldsymbol{K}}\sum_{t\in \boldsymbol{T}}\epsilon^2_{jkt}(g_{jkt}-z_{jkt})
\end{aligned} \quad (4.5)$$

$$\text{s.t. } (3.2),\ (3.3),\ (3.5),\ (3.7)-(3.15),\ (3.17),\ (3.19),\ (3.20),\ (3.25). \tag{4.6}$$

The Lagrangean subproblem decomposes into three subproblems for a given set of Lagrange multiplier values. The objective value of the first subproblem is a function of $\boldsymbol{\lambda}$, i.e. $Z_1(\boldsymbol{\lambda})$, the objective value of the second subproblem is a function of $\boldsymbol{\lambda},\ \boldsymbol{\mu}^1,\ \boldsymbol{\mu}^2,\ \boldsymbol{\delta}^1,\ \boldsymbol{\delta}^2,\ \theta\ \boldsymbol{\epsilon}^1$and $\boldsymbol{\epsilon}^1$, i.e. $Z_2(\boldsymbol{\lambda},\ \boldsymbol{\mu}^1,\ \boldsymbol{\mu}^2,\ \boldsymbol{\delta}^1,\ \boldsymbol{\delta}^2,\ \theta,\ \boldsymbol{\epsilon}^1,\ \boldsymbol{\epsilon}^2)$ and the objective value of the third subproblem is a function of $\boldsymbol{\mu}^1,\ \boldsymbol{\mu}^2,\ \boldsymbol{\delta}^1,\ \boldsymbol{\delta}^2,\ \boldsymbol{\epsilon}^1$ and $\boldsymbol{\epsilon}^2$, i.e. $Z_3(\boldsymbol{\mu}^1,\ \boldsymbol{\mu}^2,\ \boldsymbol{\delta}^1,\ \boldsymbol{\delta}^2,\ \boldsymbol{\epsilon}^1,\ \boldsymbol{\epsilon}^2)$. For simplicity, we will denote the objective function values of the first, second and third subproblems as $Z_1,\ Z_2$ and Z_3 in the sequel, respectively. These subproblems and their solution procedures are discussed in the following subsections.

4.2.1. First Subproblem

The mathematical program of the first subproblem SP_1 can be given as:

$$SP_1 : Z_1(\boldsymbol{\lambda}) = \max \quad L \quad -\sum_{t\in \boldsymbol{T}} \left(\sum_{i\in \boldsymbol{N}} \lambda_{it} f_i \right) n_t \tag{4.7}$$

$$\text{s.t. } (3.2),\ (3.3)$$

$$L \geq 0,\ n_t \in \{0,1\} \qquad t \in \boldsymbol{T} \tag{4.8}$$

The objective function can be written as $L - \sum_{t\in \boldsymbol{T}} \upsilon_t n_t$ where $\upsilon_t = \sum_{i\in \boldsymbol{N}} \lambda_{it} f_i$. Notice that υ_t values, coefficients of n_t, are nonnegative. Therefore, since the objective is a maximization we would like to assign zero to variable n_t instead of one. Suppose that $L = t_0 - 1$. When $n_t = 0$ constraints (3.3) become $t \geq t_0$. This means, we have to assign $n_t = 1$ for $t < t_0$. Hence, for a given L we can determine the n_t variables and calculate the objective function value. As a result, we can find Z_1 by enumerating objective function for all possible values of $L \in \{1, ..., T\}$ and select the solution which gives the highest objective.

Proposition 4.1. *The computational complexity of the solution procedure for SP_1 is $\mathcal{O}(NT)$.*

Proof. Calculating υ_t for a given t is $\mathcal{O}(N)$. Then calculating all υ_t values are $\mathcal{O}(NT)$. For a given L value, calculating the objective function is $\mathcal{O}(T)$. Finding L which gives the maximum objective value is $\mathcal{O}(T)$. Hence, the computational complexity of the solution procedure is $\mathcal{O}(NT)$. □

4.2.2. Second Subproblem

The mathematical program of the second subproblem SP_2 can be given as:

$$\begin{aligned} SP_2 : Z_2(\boldsymbol{\lambda},\ \boldsymbol{\mu}^1,\ \boldsymbol{\mu}^2,\ \boldsymbol{\delta}^1,\ \boldsymbol{\delta}^2,\ \theta,\ \boldsymbol{\epsilon}^1,\ \boldsymbol{\epsilon}^2) = \max\ & -\theta \sum_{j\in\boldsymbol{N}} \sum_{k\in\boldsymbol{K}} c_{jk} x_{jk} \\ & + \sum_{j\in\boldsymbol{N}} \sum_{k\in\boldsymbol{K}} \sum_{t\in\boldsymbol{T}} \left\{ \left(\sum_{i\in\boldsymbol{N}} \lambda_{it} a_{ijk} \right) + \epsilon^1_{jkt} - \epsilon^2_{jkt} \right\} z_{jkt} \\ & + \sum_{i\in\boldsymbol{N}} \sum_{j\in\boldsymbol{N}} \sum_{k\in\boldsymbol{K}} \sum_{t\in\boldsymbol{T}} \gamma_{ijkt} u_{ijkt} \end{aligned} \tag{4.9}$$

s.t. (3.10) − (3.15)

$$x_{jk}, z_{jkt}, u_{ijkt} \in \{0,1\} \qquad i,j \in \boldsymbol{N};\ k \in \boldsymbol{K};\ t \in \boldsymbol{T} \tag{4.10}$$

where

$$\gamma_{ijkt} = h_{jk}(\mu^1_{it} - \mu^2_{it}) + \sum_{v\in\boldsymbol{N}} \sum_{l\in\boldsymbol{K}} (\delta^1_{ivljkt} - \delta^1_{ijkvlt}) - \sum_{v\in\boldsymbol{N}} \delta^2_{ijkv0t}. \tag{4.11}$$

Notice that SP_2 can be decomposed with respect to point j and sensor type k. This means, $Z_2 = \sum_{j\in\boldsymbol{N}} \sum_{k\in\boldsymbol{K}} Z_2^{jk}$ where Z_2^{jk} is obtained by solving problem SP_2^{jk} given

as:

$$\begin{aligned} SP_2^{jk} : Z_2^{jk}(\boldsymbol{\lambda},\ \boldsymbol{\mu}^1,\ \boldsymbol{\mu}^2,\ \boldsymbol{\delta}^1,\ \boldsymbol{\delta}^2,\ \theta,\ \boldsymbol{\epsilon}^1,\ \boldsymbol{\epsilon}^2) = \max \quad & -\theta c_{jk} x_{jk} \\ & + \sum_{t \in \boldsymbol{T}} \left\{ \left(\sum_{i \in \boldsymbol{N}} \lambda_{it} a_{ijk} \right) + \epsilon^1_{jkt} - \epsilon^2_{jkt} \right\} z_{jkt} \\ & + \sum_{i \in \boldsymbol{N}} \sum_{t \in \boldsymbol{T}} \gamma_{ijkt} u_{ijkt} \end{aligned} \tag{4.12}$$

s.t. (3.10) − (3.15), (4.10).

A feasible solution of SP_2^{jk} assigns a sink for a deployed sensor (j,k) at the periods it is active. This means, if $x_{jk} = 0$, then $z_{jkt} = 0$ and $u_{ijkt} = 0$ necessarily. On the other hand, if we have a sensor (j,k), i.e. $x_{jk} = 1$, and if it is active at period t, i.e. $z_{jkt} = 1$, then the best possible sink to assign sensor (j,k) can be $i = \arg\max_{i \in \boldsymbol{N}} \{\gamma_{ijkt}\}$. Let $CU = \max_{i \in \boldsymbol{N}} \{\gamma_{ijkt}\}$. If there is a sensor (j,k) it can be activated at period t if

$$CZ_t = \left\{ \left(\sum_{i \in \boldsymbol{N}} \lambda_{it} a_{ijk} \right) + \epsilon^1_{jkt} - \epsilon^2_{jkt} \right\} + CU > 0. \tag{4.13}$$

Finally, it will be meaningful to deploy a sensor (j,k), i.e. $x_{jk} = 1$, if

$$CX = -\theta c_{jk} + \sum_{t \in \boldsymbol{T}} CZ_t > 0. \tag{4.14}$$

Therefore, for a given j and k indexes, we start by assuming $x_{jk} = 1$ and try to determine the periods t, in which we assign the sink $(i,0)$ from $i = \arg\max_{i \in \boldsymbol{N}} \{\gamma_{ijkt}\}$, to activate the deployed sensor (j,k) by checking (4.13). After enumarating over all points i and periods t, if (4.14) is provided then we can activate sensor (j,k) and assign the selected sink to it in the determined periods. In the case we could not satisfy (4.14), i.e. our initial assumption $x_{jk} = 1$ was not true, the solution of the subproblem (j,k) will be $x_{jk} = 0$, $z_{jkt} = 0$ and $u_{ijkt} = 0$ for all points i and periods t.

Proposition 4.2. *The computational complexity of the solution procedure for SP_2 is*

$\mathcal{O}(N^3K^2T)$.

Proof. First let us consider a given subproblem (j,k). Calculating γ_{ijkt} for all points i in a period t is $\mathcal{O}(N^2K)$. Calculating CU for a period is $\mathcal{O}(N)$. Calculating CZ_t for a period is $\mathcal{O}(N)$ and $\mathcal{O}(NT)$ for all periods. Calculating CX is $\mathcal{O}(T)$. Solving a subproblem (j,k) is $\mathcal{O}(N^2KT)$. Therefore, computational complexity of the solution procedure for SP_2 is $\mathcal{O}(N^3K^2T)$. □

4.2.3. Third Subproblem

The mathematical program of the third subproblem SP_3 can be given as:

$$\begin{aligned} SP_3 : Z_3(\boldsymbol{\mu}^1,\ \boldsymbol{\mu}^2,\ \boldsymbol{\delta}^1,\ \boldsymbol{\delta}^2,\ \boldsymbol{\epsilon}^1,\ \boldsymbol{\epsilon}^2) = \max\ & -\sum_{j\in\boldsymbol{N}}\sum_{k\in\boldsymbol{K}}\sum_{t\in\boldsymbol{T}}(\epsilon^1_{jkt}-\epsilon^2_{jkt})g_{jkt} \\ & -\sum_{j\in\boldsymbol{N}}\sum_{k\in\boldsymbol{K}}\sum_{i\in\boldsymbol{N}}\sum_{t\in\boldsymbol{T}}(\mu^1_{it}-\mu^2_{it})y_{jki0t} \\ & -\sum_{i\in\boldsymbol{N}}\sum_{l\in\boldsymbol{K}}\sum_{j\in\boldsymbol{N}}\sum_{t\in\boldsymbol{T}} M_2\left\{\sum_{k\in\boldsymbol{K}}\left(\sum_{v\in\boldsymbol{N}}\delta^1_{viljkt}\right)w_{iljkt}+\left(\sum_{v\in\boldsymbol{N}}\delta^2_{vilj0t}\right)w_{ilj0t}\right\} \end{aligned} \tag{4.15}$$

s.t. (3.9), (3.20), (3.17), (3.25)

Constraints (3.5), (3.7), (3.8) where z_{jkt} is replaced by g_{jkt} (4.16)

$g_{jkt}, w_{jkilt} \in \{0,1\}$ and $y_{jkilt} \geq 0$ $\quad i,j \in \boldsymbol{N},\ k \in \boldsymbol{K},\ l \in \boldsymbol{K'},\ t \in \boldsymbol{T}$ (4.17)

Notice that SP_3 is a MILP. Hence, it can be solved using a commercial LP solver such as CPLEX (ilog, 2007).

Hence, we obtain an upper bound on the network lifetime L with the Lagrangean

problem as:

$$L_{UB} = Z_1 + Z_2 + Z_3 + \Delta \tag{4.18}$$

where

$$\Delta = \theta B + \sum_{v\in \boldsymbol{N}} \sum_{i\in \boldsymbol{N}} \sum_{l\in \boldsymbol{K}} \sum_{j\in \boldsymbol{N}} \sum_{t\in \boldsymbol{T}} \left\{ \left(\sum_{k\in \boldsymbol{K}} \delta^1_{viljkt} M_2 \right) + \delta^2_{vilj0t} (M_2 + \mathbf{1}_j(v) x_{j0}) \right\} \tag{4.19}$$

An upper bound L_{UB} for PSRPC can be obtained with a given set of Lagrange multipliers $\{\boldsymbol{\lambda},\ \boldsymbol{\mu}^1,\ \boldsymbol{\mu}^2,\ \boldsymbol{\delta}^1,\ \boldsymbol{\delta}^2,\ \theta,\ \boldsymbol{\epsilon}^1,\ \boldsymbol{\epsilon}^2\}$ as explained above. The best, i.e. smallest, upper bound can be found by solving the following Lagrangean dual problem

$$L^*_{UB} = \min_{\boldsymbol{\lambda}\geq 0, \boldsymbol{\mu}^1\geq 0, \boldsymbol{\mu}^2\geq 0, \boldsymbol{\delta}^1\geq 0, \boldsymbol{\delta}^2\geq 0, \theta\geq 0, \boldsymbol{\epsilon}^1\geq 0, \boldsymbol{\epsilon}^2\geq 0} L_{UB}(\boldsymbol{\lambda},\ \boldsymbol{\mu}^1,\ \boldsymbol{\mu}^2,\ \boldsymbol{\delta}^1,\ \boldsymbol{\delta}^2,\ \theta,\ \boldsymbol{\epsilon}^1,\ \boldsymbol{\epsilon}^2) \tag{4.20}$$

using the subgradient optimization algorithm (Held et al., 1974). At each iteration r of the subgradient optimization procedure, the current upper bound $L^{(r)}_{UB}$ is obtained by solving $SP_1,\ SP_2$ and SP_3 to optimality. We have relaxed some of the constraints of PSRPC in the Lagrangean subproblem. Therefore, for a feasible solution of the Lagrangean subproblem at iteration r, say $(L^{(r)}, \mathbf{n}^{(r)}, \mathbf{x}^{(r)}, \mathbf{z}^{(r)}, \mathbf{g}^{(r)}, \mathbf{u}^{(r)}, \mathbf{y}^{(r)}, \mathbf{w}^{(r)})$, the nonnegative deviation from feasibility with respect to relaxed constraints can be represented with subgradients which are defined as:

$$SG_{it}^{\lambda} = \left(\sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} a_{ijk} z_{jkt}^{(r)} \right) - f_i n_t^{(r)} \qquad i \in \boldsymbol{N};\ t \in \boldsymbol{T} \tag{4.21}$$

$$SG_{it}^{\mu^1} = \sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} \left(h_{jk} u_{ijkt}^{(r)} - y_{jki0t}^{(r)} \right) \qquad i \in \boldsymbol{N};\ t \in \boldsymbol{T} \tag{4.22}$$

$$SG_{it}^{\mu^2} = \sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} \left(y_{jki0t}^{(r)} - h_{jk} u_{ijkt}^{(r)} \right) \qquad i \in \boldsymbol{N};\ t \in \boldsymbol{T} \tag{4.23}$$

$$SG_{viljkt}^{\delta^1} = M_2(1 - w_{iljkt}^{(r)}) - u_{vilt}^{(r)} + u_{vjkt}^{(r)} \qquad v, i, j \in \boldsymbol{N};\ k, l \in \boldsymbol{K};\ t \in \boldsymbol{T} \tag{4.24}$$

$$SG_{vilj0t}^{\delta^2} = M_2(1 - w_{ilj0t}^{(r)}) - u_{vilt}^{(r)} + \mathbf{1}_j(v) x_{j0}^{(r)} \qquad v, i, j \in \boldsymbol{N};\ l \in \boldsymbol{K};\ t \in \boldsymbol{T} \tag{4.25}$$

$$SG^{\theta} = B - \sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} c_{jk} x_{jk}^{(r)} \tag{4.26}$$

$$SG_{jkt}^{\epsilon^1} = z_{jkt}^{(r)} - g_{jkt}^{(r)} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K};\ t \in \boldsymbol{T} \tag{4.27}$$

$$SG_{jkt}^{\epsilon^2} = g_{jkt}^{(r)} - z_{jkt}^{(r)} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K};\ t \in \boldsymbol{T}. \tag{4.28}$$

They are used to update the Lagrange multipliers by defining a step length $\xi^{(r)}$ as

$$\xi^{(r)} = \frac{\pi (L_{UB}^{(r)} - L_{LB}^{*})}{A} \tag{4.29}$$

at each iteration r. Here L_{LB}^{*} is the best available lower bound, π is step length parameter and A is calculated as:

$$\begin{aligned} A = & \sum_{i \in \boldsymbol{N}} \sum_{t \in \boldsymbol{T}} \left((SG_{it}^{\lambda})^2 + (SG_{it}^{\mu^1})^2 + (SG_{it}^{\mu^2})^2 \right) \\ & + \sum_{v \in \boldsymbol{N}} \sum_{i \in \boldsymbol{N}} \sum_{l \in \boldsymbol{K}} \sum_{j \in \boldsymbol{N}} \sum_{t \in \boldsymbol{T}} \left\{ \left(\sum_{k \in \boldsymbol{K}} (SG_{viljkt}^{\delta^1})^2 \right) + (SG_{vilj0t}^{\delta^2})^2 \right\} + (SG^{\theta})^2 \\ & + \sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} \sum_{t \in \boldsymbol{T}} \left((SG_{jkt}^{\epsilon^1})^2 + (SG_{jkt}^{\epsilon^2})^2 \right) \end{aligned} \tag{4.30}$$

Then, Lagrange multipliers $\boldsymbol{\lambda}^{(r)},\ \boldsymbol{\mu}^{1(r)},\ \boldsymbol{\mu}^{2(r)},\ \boldsymbol{\delta}^{1(r)},\ \boldsymbol{\delta}^{2(r)},\ \theta^{(r)},\ \boldsymbol{\epsilon}^{1(r)},\ \boldsymbol{\epsilon}^{2(r)}$ at iteration

r are updated as:

$$\lambda_{it}^{(r+1)} = \max\{\lambda_{it}^{(r)} - \xi^{(r)}(SG_{it}^{\lambda}),\ 0\} \qquad i \in \boldsymbol{N};\ t \in \boldsymbol{T} \tag{4.31}$$

$$\mu_{it}^{1(r+1)} = \max\{\mu_{it}^{1(r)} - \xi^{(r)}(SG_{it}^{\mu^1}),\ 0\} \qquad i \in \boldsymbol{N};\ t \in \boldsymbol{T} \tag{4.32}$$

$$\mu_{it}^{2(r+1)} = \max\{\mu_{it}^{2(r)} - \xi^{(r)}(SG_{it}^{\mu^2}),\ 0\} \qquad i \in \boldsymbol{N};\ t \in \boldsymbol{T} \tag{4.33}$$

$$\delta_{viljkt}^{1(r+1)} = \max\{\delta_{viljkt}^{1(r)} - \xi^{(r)}(SG_{viljkt}^{\delta^1}),\ 0\} \qquad v,i,j \in \boldsymbol{N};\ k,l \in \boldsymbol{K};\ t \in \boldsymbol{T} \tag{4.34}$$

$$\delta_{vilj0t}^{2(r+1)} = \max\{\delta_{vilj0t}^{2(r)} - \xi^{(r)}(SG_{vilj0t}^{\delta^2}),\ 0\} \qquad v,i,j \in \boldsymbol{N};\ l \in \boldsymbol{K};\ t \in \boldsymbol{T} \tag{4.35}$$

$$\theta^{(r+1)} = \max\{\theta^{(r)} - \xi^{(r)}(SG^{\theta}),\ 0\} \tag{4.36}$$

$$\epsilon_{jkt}^{1(r+1)} = \max\{\epsilon_{jkt}^{1(r)} - \xi^{(r)}(SG_{jkt}^{\epsilon^1}),\ 0\} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K};\ t \in \boldsymbol{T} \tag{4.37}$$

$$\epsilon_{jkt}^{2(r+1)} = \max\{\epsilon_{jkt}^{2(r)} - \xi^{(r)}(SG_{jkt}^{\epsilon^2}),\ 0\} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K};\ t \in \boldsymbol{T}. \tag{4.38}$$

Lower bounds $L_{LB}^{(r)}$ are computed at each iteration r by constructing a feasible solution to PSRPC from the solution of the Lagrangean subproblem. Feasible solution procedures are explained in section (4.4). The output of the first Lagrangean Heuristic (LH_1) is the best lower bound L_{LB}^* found from the feasible solution generation procedure and the smallest upper bound L_{UB}^* found during the iterations. There are different alternatives to finalize the iterations of the Lagrangean heuristic. It is possible to terminate the subgradient algorithm if the gap between the best upper and lower bounds is less than a certain threshold value, i.e. $L_{UB}^* - L_{LB}^* < \eta_1$ where η_1 is a small nonnegative constant. However, this criterion may not be succesful if there is a duality gap larger than η_1. Therefore, we can consider to use the step length parameter π as another termination criterion. The value of π is halved if there is no improvement in the best upper bound L_{UB}^* for consecutive κ iterations. The value $\kappa = 20$ is suggested in the literature (Beasley, 1993). We terminate the subgradient iterations if π becomes smaller than a threshold value η_2. Related with the performance of the Lagrangean heuristic, we observe that the feasible solution generation procedures find good lower bounds in early iterations whereas the Lagrangean subproblem cannot improve the best upper bound value considerably after some number of iterations. Therefore, in order to save time the number of iterations in subgradient algorithm is limited with parameter *iterlim*. The steps of LH_1 are given in Figure (4.1).

1. Initialization: Set iteration counter $r = 0$, $\pi^{(0)} = 2$, $L^*_{UB} = \infty$, $L^*_{LB} = 0$ and $\lambda^{(r)}_{it}$, $\mu^{1(r)}_{it}$, $\mu^{2(r)}_{it}$, $\delta^{1(r)}_{viljkt}$, $\delta^{2(r)}_{vilj0t}$, $\theta^{(r)}$, $\epsilon^{1(r)}_{jkt}$, $\epsilon^{2(r)}_{jkt}$ for $v, i, j \in \boldsymbol{N}$, $k, l \in \boldsymbol{K}$, $t \in \boldsymbol{T}$.
2. While $L^*_{UB} - L^*_{LB} \geq \eta_1$ and $\pi \geq \eta_2$ and $r \leq iterlim$ Do
3. Solve subproblems SP_1, SP_2, SP_3,
   compute $L_{UB} = Z_1 + Z_2 + Z_3 + \Delta$
   and update $L_{UB} = \min\{L^*_{UB}, L^{(r)}_{UB}\}$.
4. If L^*_{UB} is not updated consecutive last κ iterations Then set $\pi \leftarrow \pi/2$.
5. Construct a feasible solution with objective value $L^{(r)}_{LB}$ using one of the algorithms described in section (4.4) and update $L^*_{LB} = \max\{L^*_{LB}, L^{(r)}_{LB}\}$.
6. Update Lagrange multipliers $\boldsymbol{\lambda}$, $\boldsymbol{\mu}^1$, $\boldsymbol{\mu}^2$, $\boldsymbol{\delta}^1$, $\boldsymbol{\delta}^2$, θ, $\boldsymbol{\epsilon}^1$, $\boldsymbol{\epsilon}^2$ with equations (4.31) - (4.38).
7. $r \leftarrow r + 1$
8. End While

Figure 4.1. First Lagrangean heuristic, LH_1

Proposition 4.3. *The computational complexity of the solution procedure for LH_1 is $\mathcal{O}(N^3K^2T(iterlim) + \mathcal{O}(O_{SP_3})(iterlim) + O_{LB}(iterlim))$ where $\mathcal{O}(O_{SP_3})$ denotes the complexity of the algorithm to solve SP_3 and $\mathcal{O}(O_{LB})$ represents the complexity of the algorithm to generate a lower bound.*

Proof. Initializing the algorithm is $\mathcal{O}(N^3K^2T)$. Solving SP_1 is $\mathcal{O}(NT)$ and SP_2 is $\mathcal{O}(N^3K^2T)$. Solving SP_3 is $\mathcal{O}(O_{SP_3})$. Constructing a lower bound is $\mathcal{O}(O_{LB})$. Updating Lagrange multipliers is $\mathcal{O}(N^3K^2T)$. Then, computational complexity of the algorithm is $\mathcal{O}(N^3K^2T(iterlim) + \mathcal{O}(O_{SP_3})(iterlim) + O_{LB}(iterlim))$. □

4.3. Second Lagrangean Heuristic

Second Lagrangean heuristic is based on the relaxation of the constraints (3.4), (4.3), (4.4), (3.22), (3.23) and (3.18) with multipliers $\boldsymbol{\lambda} \geq 0,\ \boldsymbol{\mu}^1 \geq 0,\ \boldsymbol{\mu}^2 \geq 0,\ \boldsymbol{\delta}^1 \geq 0,\ \boldsymbol{\delta}^2 \geq 0$ and $\theta \geq 0$, respectively. Then, the following Lagrangean subproblem is obtained:

$$
\begin{aligned}
L_{UB}(\boldsymbol{\lambda},\, \boldsymbol{\mu}^1,\ \boldsymbol{\mu}^2,\ \boldsymbol{\delta}^1,\ \boldsymbol{\delta}^2,\ \theta) = \max L + \ \theta \left(B - \sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} c_{jk} x_{jk} \right) \\
+ \sum_{i \in \boldsymbol{N}} \sum_{t \in \boldsymbol{T}} \mu^1_{it} \left\{ \sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} (h_{jk} u_{ijkt} - y_{jki0t}) \right\} \\
+ \sum_{i \in \boldsymbol{N}} \sum_{t \in \boldsymbol{T}} \mu^2_{it} \left\{ \sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} (y_{jki0t} - h_{jk} u_{ijkt}) \right\} \\
+ \sum_{v \in \boldsymbol{N}} \sum_{i \in \boldsymbol{N}} \sum_{l \in \boldsymbol{K}} \sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} \sum_{t \in \boldsymbol{T}} \delta^1_{viljkt} \left(M_2(1 - w_{iljkt}) - u_{vilt} + u_{vjkt} \right) \\
+ \sum_{v \in \boldsymbol{N}} \sum_{i \in \boldsymbol{N}} \sum_{l \in \boldsymbol{K}} \sum_{j \in \boldsymbol{N}} \sum_{t \in \boldsymbol{T}} \delta^2_{vilj0t} \left(M_2(1 - w_{ilj0t}) - u_{vilt} + \mathbf{1}_j(v) x_{j0} \right) \\
+ \sum_{i \in \boldsymbol{N}} \sum_{t \in \boldsymbol{T}} \lambda_{it} \left\{ \left(\sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} a_{ijk} z_{jkt} \right) - f_i n_t \right\}
\end{aligned}
\tag{4.39}
$$

$$
\text{such that (3.2),\ (3.3),\ (3.5),\ (3.7) – (3.15),\ (3.17),\ (3.19),\ (3.20),\ (3.25).} \tag{4.40}
$$

The Lagrangean subproblem decomposes into two subproblems for a given set of Lagrange multipliers. The objective value of the first subproblem is a function of $\boldsymbol{\lambda}$, i.e. $Z_1(\boldsymbol{\lambda})$, and the objective value of the second subproblem is a function of $\boldsymbol{\lambda},\ \boldsymbol{\mu}^1,\ \boldsymbol{\mu}^2,\ \boldsymbol{\delta}^1,\ \boldsymbol{\delta}^2$ and θ, i.e. $Z_2(\boldsymbol{\lambda},\ \boldsymbol{\mu}^1,\ \boldsymbol{\mu}^2,\ \boldsymbol{\delta}^1,\ \boldsymbol{\delta}^2,\ \theta)$. For simplicity, we will denote the objective function values of the first and second subproblems as Z_1 and Z_2 in the sequel.

4.3.1. First Subproblem

The mathematical program of the first subproblem SP_1 is the same as the first subproblem given in Subsection 4.2.1. Therefore, the solution strategy is the same as SP_1's.

4.3.2. Second Subproblem

The mathematical program of the second subproblem SP_2 can be given as:

$$\begin{aligned} SP_2 : Z_2(\boldsymbol{\lambda},\ \boldsymbol{\mu}^1,\ \boldsymbol{\mu}^2,\ \boldsymbol{\delta}^1,\ \boldsymbol{\delta}^2,\ \theta) = \max\ & -\theta \sum_{j\in \boldsymbol{N}} \sum_{k\in \boldsymbol{K}} c_{jk} x_{jk} \\ & + \sum_{j\in \boldsymbol{N}} \sum_{k\in \boldsymbol{K}} \sum_{t\in \boldsymbol{T}} \left(\sum_{i\in \boldsymbol{N}} \lambda_{it} a_{ijk} \right) z_{jkt} + \sum_{i\in \boldsymbol{N}} \sum_{j\in \boldsymbol{N}} \sum_{k\in \boldsymbol{K}} \sum_{t\in \boldsymbol{T}} \gamma_{ijkt} u_{ijkt} \\ & - \sum_{i\in \boldsymbol{N}} \sum_{j\in \boldsymbol{N}} \sum_{k\in \boldsymbol{K}} \sum_{t\in \boldsymbol{T}} (\mu^1_{it} - \mu^2_{it}) y_{jki0t} \\ & - \sum_{v\in \boldsymbol{N}} \sum_{i\in \boldsymbol{N}} \sum_{l\in \boldsymbol{K}} \sum_{j\in \boldsymbol{N}} \sum_{t\in \boldsymbol{T}} M_2 \left(\left(\sum_{k\in \boldsymbol{K}} \delta^1_{viljkt} w_{iljkt} \right) + \delta^2_{vilj0t} w_{ilj0t} \right) \end{aligned} \tag{4.41}$$

s.t. (3.5), (3.7)- (3.15), (3.17), (3.19), (3.20), (3.25)

$$x_{jk}, z_{jkt}, u_{ijkt}, w_{jkilt} \in \{0,1\} \text{ and } y_{jkilt} \geq 0 \qquad i,j \in \boldsymbol{N},\ k \in \boldsymbol{K},\ l \in \boldsymbol{K'},\ t \in \boldsymbol{T} \tag{4.42}$$

where γ_{ijkt} is as defined in equation (4.11).

Notice that SP_2 can decompose with respect to point j. This means, $Z_2 =$

$\sum_{j\in \boldsymbol{N}} Z_2^j$ where Z_2^j is obtained by solving problem SP_2^j given as:

$$\begin{aligned} SP_2^j : Z_2^j(\boldsymbol{\lambda},\ \boldsymbol{\mu}^1,\ \boldsymbol{\mu}^2,\ \boldsymbol{\delta}^1,\ \boldsymbol{\delta}^2,\ \theta) = \max\ & -\theta \sum_{k\in \boldsymbol{K}} c_{jk}x_{jk} \\ & + \sum_{k\in \boldsymbol{K}} \sum_{t\in \boldsymbol{T}} \left(\sum_{i\in \boldsymbol{N}} \lambda_{it} a_{ijk} \right) z_{jkt} + \sum_{i\in \boldsymbol{N}} \sum_{k\in \boldsymbol{K}} \sum_{t\in \boldsymbol{T}} \gamma_{ijkt} u_{ijkt} \\ & - \sum_{i\in \boldsymbol{N}} \sum_{k\in \boldsymbol{K}} \sum_{t\in \boldsymbol{T}} (\mu_{it}^1 - \mu_{it}^2) y_{jki0t} \\ & - \sum_{v\in \boldsymbol{N}} \sum_{i\in \boldsymbol{N}} \sum_{l\in \boldsymbol{K}} \sum_{t\in \boldsymbol{T}} M_2 \left(\left(\sum_{k\in \boldsymbol{K}} \delta_{viljkt}^1 w_{iljkt} \right) + \delta_{vilj0t}^2 w_{ilj0t} \right) \end{aligned} \tag{4.43}$$

$$\text{s.t. } (3.5),\ (3.7)-(3.15),\ (3.17),\ (3.19),\ (3.20),\ (3.25),\ (4.10).$$

Notice that SP_2^j is a MILP. Hence, it can be solved with the help of a commercial LP solver such as CPLEX (ilog, 2007).

Hence, we get an upperbound on the network lifetime L with first Lagrangean subproblem as:

$$L_{UB} = Z_1 + Z_2 + \Delta \tag{4.44}$$

where Δ is as defined in equation (4.19).

Similar to the first Lagrangean heuristic, we try to find the values of multipliers $\{\boldsymbol{\lambda},\ \boldsymbol{\mu}^1,\ \boldsymbol{\mu}^2,\ \boldsymbol{\delta}^1,\ \boldsymbol{\delta}^2,\ \theta\}$ which gives the best upper bound L_{UB} for PSRPC. For this purpose, we solve the Lagrangean dual problem

$$L_{UB}^* = \min_{\boldsymbol{\lambda}\geq 0, \boldsymbol{\mu}^1\geq 0, \boldsymbol{\mu}^2\geq 0, \boldsymbol{\delta}^1\geq 0, \boldsymbol{\delta}^2\geq 0, \theta\geq 0} L_{UB}(\boldsymbol{\lambda},\ \boldsymbol{\mu}^1,\ \boldsymbol{\mu}^2,\ \boldsymbol{\delta}^1,\ \boldsymbol{\delta}^2,\ \theta) \tag{4.45}$$

using the subgradient optimization algorithm. At each iteration r of the subgradient algorithm, the current upper bound $L_{UB}^{(r)}$ is obtained by solving SP_1 and SP_2

1. Initialization: Set iteration counter $r = 0$, $\pi^{(0)} = 2$, $L^*_{UB} = \infty$, $L^*_{LB} = 0$ and $\lambda^{(r)}_{it}$, $\mu^{1(r)}_{it}$, $\mu^{2(r)}_{it}$, $\delta^{1(r)}_{viljkt}$, $\delta^{2(r)}_{vilj0t}$, $\theta^{(r)}$ for $v, i, j \in \boldsymbol{N}$, $k, l \in \boldsymbol{K}$, $t \in \boldsymbol{T}$.
2. While $L^*_{UB} - L^*_{LB} \geq \eta_1$ and $\pi \geq \eta_2$ and $r \leq iterlim$ Do
3. Solve subproblems SP_1 and SP_2,
   compute $L_{UB} = Z_1 + Z_2 + Z_3 + \Delta$
   and update $L_{UB} = \min\{L^*_{UB},\ L^{(r)}_{UB}\}$.
4. If L^*_{UB} is not updated consecutive last κ iterations Then set $\pi \leftarrow \pi/2$.
5. Construct a feasible solution with objective value $L^{(r)}_{LB}$ using one of the algorithms described in section (4.4) and update $L^*_{LB} = \max\{L^*_{LB},\ L^{(r)}_{LB}\}$.
6. Update Lagrange multipliers $\boldsymbol{\lambda}$, $\boldsymbol{\mu}^1$, $\boldsymbol{\mu}^2$, $\boldsymbol{\delta}^1$, $\boldsymbol{\delta}^2$, θ with equations (4.31) - (4.36).
7. $r \leftarrow r + 1$
8. End While

Figure 4.2. Second Lagrangean heuristic, LH_2

to optimality. For a feasible solution of the Lagrangean subproblem at iteration r, say $(L^{(r)}, \mathbf{n}^{(r)}, \mathbf{x}^{(r)}, \mathbf{z}^{(r)}, \mathbf{u}^{(r)}, \mathbf{y}^{(r)}, \mathbf{w}^{(r)})$, subgradients can be calculated as in equations (4.21) - (4.26) given in the previous section. The step length is determined with the equation (4.29) where A value is found as:

$$A = \sum_{i\in\boldsymbol{N}}\sum_{t\in\boldsymbol{T}}\left((SG^{\lambda}_{it})^2 + (SG^{\mu^1}_{it})^2 + (SG^{\mu^2}_{it})^2\right) + \sum_{v\in\boldsymbol{N}}\sum_{i\in\boldsymbol{N}}\sum_{l\in\boldsymbol{K}}\sum_{j\in\boldsymbol{N}}\sum_{t\in\boldsymbol{T}}\left\{\left(\sum_{k\in\boldsymbol{K}}(SG^{\delta^1}_{viljkt})^2\right) + (SG^{\delta^2}_{vilj0t})^2\right\} + (SG^{\theta})^2 \tag{4.46}$$

Finally, Lagrange multipliers $\boldsymbol{\lambda}^{(r)}$, $\boldsymbol{\mu}^{1(r)}$, $\boldsymbol{\mu}^{2(r)}$, $\boldsymbol{\delta}^{1(r)}$, $\boldsymbol{\delta}^{2(r)}$, $\theta^{(r)}$ at iteration r can be updated with the equations (4.31) - (4.36). We can summarize the steps of LH_2 as in Figure (4.2).

Proposition 4.4. *The computational complexity of the solution procedure for* LH_2 *is* $\mathcal{O}(O_{SP_2}(iterlim) + O_{LB}(iterlim) + N^3K^2T(iterlim))$ *where* $\mathcal{O}(O_{SP_2})$ *indicates the complexity of the algorithm to solve* SP_2 *and* $\mathcal{O}(O_{LB})$ *represents the complexity of the algorithm to generate a lower bound.*

Proof. Initializing the algorithm is $\mathcal{O}(N^3K^2T)$. Solving SP_1 is $\mathcal{O}(NT)$. Solving SP_2 is $\mathcal{O}(O_{SP_2})$. Constructing a lower bound is $\mathcal{O}(O_{LB})$. Updating Lagrange multipliers is $\mathcal{O}(N^3K^2T)$. Then, complexity of the algorithm is $\mathcal{O}(O_{SP_2}(iterlim) + O_{LB}(iterlim) + N^3K^2T(iterlim))$. □

4.4. Generating a Feasible Solution

The algorithms for Lagrangean heuristics given in sections (4.2) and (4.3) require lower bounds in order to update the Lagrange multipliers at each iteration. The lower bounds can be obtained by a heuristic that uses the current solution of the Lagrangean subproblem. The heuristic constructs a feasible solution from it by recovering the infeasibilities with respect to the relaxed constraints of the Lagrangean subproblem. In this section, we introduce two different algorithms to generate a feasible solution out of the current solution of the Lagrangean subproblem.

4.4.1. Greedy Heuristic

First heuristic consists of three main steps, namely providing feasibility subject to coverage and budget constraints, providing feasibility subject to sink assignment constraints and determining feasible values for variables $\mathbf{y}$ and $\mathbf{w}$. Each step of the algorithm is done for all periods $t \leq T$. The algorithm is given in Figure (4.3).

In the solution of the Lagrangean subproblem, it is possible for a sensor (j, k) to overuse its battery energy E_k. In Step 1 of the algorithm, we find the first period $\bar{t}$ that a sensor (j, k) violates the energy constraint (3.7). Then, the sensor is scheduled to be in standby mode for the periods $t \geq \bar{t}$.

We use the algorithms given in Figure (4.4) and Figure (4.5) to satisfy the coverage and budget constraints. For each period $t \leq T$, we check if every point in the sensor field is covered by the required number of sensors or not. If each point is covered and budget constraint is satisfied then we move to the next period. If budget constraint is not held, we consider to remove some of the deployed sensors without harming the coverage constraints. For this purpose, we delete sensors, i.e. set $x_{jk} = 0$, that are in standby mode until the current period since they do not contribute to the coverage of the points in any of the periods. Deleting sensor (j, k) improves remaining budget B^{rem} by c_{jk} monetary units and we continue with the process while the remaining budget is negative or we cannot find a sensor to delete.

1. For each sensor (j,k) Do

   1.1. For each $t \leq T$ Do

   If $\sum_{\bar{t}\in\{1,\ldots,t\}}(e_k^s z_{jk\bar{t}} + e_k^r \sum_{i\in\boldsymbol{N},\ l\in\boldsymbol{K}} y_{iljk\bar{t}} + e_k^c \sum_{i\in\boldsymbol{N},\ l\in\boldsymbol{K}}, y_{jkil\bar{t}}) > E_k$ Then

   $\bar{t} = t$ and Go to Step (1.2)

   End If

   End For

   1.2. For each $\hat{t} \geq \bar{t}$

   Set $z_{jk\hat{t}} = 0$

   End For

   End For

2. Use the algorithm in Figures (4.4) and (4.5) to obtain a feasible solution subject to the budget constraint and coverage constraints with objective value $\bar{L}$.

3. If $\bar{L} = 0$ Then

   Stop

   Else

   For all $t \leq \bar{L}$ Do

   Use the algorithm in Figure (4.6) to generate a feasible solution subject to the sink assignment constraints with objective value $\hat{L}$.

   End For

   End If

4. If $\hat{L} = 0$ Then

   Stop

   Else

   Solve RP described in this subsection to determine data flows $\mathbf{y}$ and dummy variables $\mathbf{w}$. The feasible solution is optimal with $L_{LB} = \hat{L}$.

   End If

Figure 4.3. Greedy heuristic, *GH*

1. Set $L_{LB}^{(r)} = T$
2. For all $t \leq L_{LB}^{(r)}$ Do

    2.1. If every point is covered Then

        If budget constraint is satisfied Then

            $t \leftarrow t + 1$

        Else /* Budget is violated */

            Remove standby sensors in periods $[0, t]$ starting with the ones having the largest cost until budget constraint is satisfied.

            If this is not possible Then

            $\bar{L} = t - 1$ and Stop

            End If

        End If

Figure 4.4. Providing feasibility subject to coverage and budget constraints (first part)

After this procedure, it is possible that the budget constraint is violated. In this case, one can consider to delete active sensors whose removal do not harm the coverage of the points. This strategy is used only when we are in the first period, i.e. $t = 1$, for the sake of simplicity of the heuristic. If budget is still violated then we set $\bar{L} = t - 1$ and stop the algorithm. On the other hand, if there is an undercovered point in the sensor field then we first try to ensure coverage in the network by activating the existing sensors. Let $\{\mathbf{L^{(r)}}, \mathbf{n^{(r)}}, \mathbf{x^{(r)}}, \mathbf{z^{(r)}}, (\mathbf{g^{(r)}}), \mathbf{y^{(r)}}, \mathbf{w^{(r)}}\}$ be the solution found with the Lagrangean subproblem at iteration r. Observe that activating a standby sensor does not demand budget usage.

2.2. Else /* There is at least one undercovered point */

2.2.1 While there is a sensor (j,k) with positive CEP_{jk} value Do

Find a sensor that can cover some undercovered points with the highest positive CEP_{jk} value and activate this sensor.

If there is no such sensor Then Go to Step (2.2.2)

End While

2.2.2 While there is a sensor (j,k) with positive CCR_{jk} value Do

Find a sensor that can cover some undercovered points with the highest positive CCR_{jk} value.

If there is no such sensor Then $\bar{L} = t - 1$ and Stop

Else

If budget is enough Then Deploy and activate this sensor.

Else

Remove standby sensors in periods $[0,t]$ starting with the ones having the largest cost until enough budget is obtained.

If budget is enough Then Deploy and activate this sensor.

Else $\bar{L} = t - 1$ and Stop

End While

End If

End For

Figure 4.5. Providing feasibility subject to coverage and budget constraints (second part)

We choose the standby sensor to activate in a gereedy way by calculating a product for each sensor in period t. For this product we first calculate the shortages in the coverage qualities for each point in the sensor field, namely the undercoverage values as:

$$U_i^{(r)} = \max\left\{ f_i n_t^{(r)} - \left(\sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} a_{ijk} z_{jkt}^{(r)} \right), 0 \right\} \qquad i \in \boldsymbol{N} \quad (4.47)$$

For a standby sensor, it is a reason of choice if it can cover as many points as possible that have positive undercoverage values. In addition, as the sensor has more remaining energy in its battery, the need for the deployment of the new sensors will be less in the future since we can activate the sensor in these periods also. Therefore, we can define the coverage energy product, i.e. $CEP_{jk}^{(r)}$, for each sensor in period t as:

$$CEP_{jk}^{(r)} = \frac{\left(\sum_{i \in \{i | U_i^{(r)} > 0\}} a_{ijk} \right) x_{jk}^{(r)} E_{jkt}^{rem}}{c_{jk}} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K} \quad (4.48)$$

where E_{jkt}^{rem} represents the remaining energy of a sensor (j, k) in period t.

Then, we activate the standby sensors starting from the one with the highest positive $CEP_{jk}^{(r)}$ value and continue until we provide coverage constraints or we do not have stanby sensors with positive $CEP_{jk}^{(r)}$ value. This means, it is possible that we can not satisfy the coverage constraints after this process. If this is the case, then we consider to deploy new sensors and activate them, i.e. $x_{jk} = z_{jkt} = 1$, in period t. In order to determine which sensors to deploy and activate in period t, we calculate a coverage cost ratio, i.e. $CCR_{jk}^{(r)}$, for each sensor (j, k) as:

$$CCR_{jk}^{(r)} = \frac{\left(\sum_{i \in \{i | U_i^{(r)} > 0\}} a_{ijk} \right) \left(1 - x_{jk}^{(r)} \right) E_k}{c_{jk}} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K} \quad (4.49)$$

where E_k represents the initial battery energy of a type-k sensor.

Observe that, we have to have sufficient budget to deploy a sensor. Then, as far as our budget allows we continue with the procedure starting from the sensor with the highest positive $CCR_{jk}^{(r)}$ value until we satisfy coverage constraints or there is no sensor with positive $CCR_{jk}^{(r)}$ value. If we can not achieve the coverage constraints and still have some candidate sensors with positive $CCR_{jk}^{(r)}$ value after the process, we try to generate sufficient budget by deleting standby sensors that are not used until the current period. The procedure proceeds as the budget allows and there are candidate sensors to deploy. If coverage is not provided, then we set $\bar{L} = t - 1$ and stop the algorithm.

Proposition 4.5. *The computational complexity of the algorithms given in Figure (4.4) and Figure (4.5) is given as* $\mathcal{O}(\tau_1 N^3 KT + \tau_1 N^2 KT^2)$ *where* $\tau_1 = \max_i \left\{ \sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} a_{ijk} \right\}$ *is the maximum number of points in the sensor field that a sensor can cover.*

Proof. Checking coverage for all points in the sensor field is $\mathcal{O}(\tau_1 N)$. Checking budget constraint is $\mathcal{O}(NK)$. In a period t, removing standby sensors until period t is $\mathcal{O}(NKT)$. Then, step (2.1) is $\mathcal{O}(\tau_1 N + NKT)$. Calculating CEP_{jkt} values in a period t is $\mathcal{O}(N^2 K)$ and finding the highest one is $\mathcal{O}(NK)$. Then, step (2.2.1) is $\mathcal{O}(\tau_1 N^3 K)$. Calculating CCR_{jkt} values in a period t is $\mathcal{O}(N^2 K)$ and finding the highest one is $\mathcal{O}(NK)$ results step (2.2.2) is $\mathcal{O}(\tau_1 N^3 K + \tau_1 N^2 KT)$. This means, step (2.2) is $\mathcal{O}(\tau_1 N^3 K + \tau_1 N^2 KT)$. As a result, providing feasibility subject to coverage and budget constraints is $\mathcal{O}(\tau_1 N^3 KT + \tau_1 N^2 KT^2)$. □

1. Initialization: For all sink $(i,0)$ set $sensorset_{i0t} = 0$ and for all sensor (j,k) set $sensorset_{jkt} = -1$. Set $setlevel = 0$.
2. For all active sensors in period t Do
   2.1. If $setlevel = 0$ Then
        For all sinks $(i,0)$ and active sensors (j,k) Do
            If $b_{jki} = 1$ and sensor (j,k) has no sink assignment Then
            $u_{ijkt} = 1,\ sensorset_{jkt} = 1$
            End If
        End For
   2.2. Else /* $setlevel \geq 1$ */
        For all disjoint active sensors (i,l) and (j,k) Do
            If $sensorset_{ilt} = setlevel,\ b_{jki} = 1$ and sensor (j,k) has no sink assignment Then
            $u_{vjkt} \leftarrow u_{vilt}$ for all v and $sensorset_{jkt} = setlevel + 1$
            End If
        End For
    End If
    $setlevel \leftarrow setlevel + 1$
   End For
3. For all active sensor (j,k)
   3.1. If sensor (j,k) has no sink assignment Then
        Use the algorithm in Figure (4.7) to generate a feasible solution subject to connectivity restrictions.
            If this is not possible Then $\hat{L} = t - 1$ and Stop
    End If
   End For

Figure 4.6. Providing feasibility in period t subject to sink assignment constraints

For each period t we use the algorithms given in Figure (4.6) and Figure (4.7) to assign a unique sink for each active sensor (j,k). The algorithm collects sensors that have similar communication properties in a set. The variable *setlevel* is an iteration counter over all sets and $sensorset_{jkt}$ represents the set index to which sensor (j,k) belongs. By definition all sinks $(i,0)$ are collected in a set at $setlevel = 0$, i.e. $sensorset_{i0t} = 0$. Then, for $setlevel = 0$ we collect all active sensors (j,k) that can communicate with at least one of the sinks $(i,0)$ directly, i.e. $b_{jki} = 1$, in a new set. This new set has the index $(setlevel + 1)$ and for all active sensors (j,k) with $b_{jki} = 1$ is in this set, i.e. $sensorset_{jkt} = setlevel + 1$. Generalizing this idea, a set with index $(setlevel + 1)$ is obtained by collecting the active sensors (j,k) that can communicate with at least one of the sensors (i,l) directly, means $b_{jki} = 1$, that is in the set with index *setlevel*. Defining $sensorset_{jkt} = setlevel + 1$ indicates sensor (j,k) is a member of the set with index $(setlevel + 1)$.

Observe that, from the criteria used for defining the sets, there is a sensor (i,l) in a set with index *setlevel* to which a flow can be sent from a sensor (j,k) in a set with index $(setlevel + 1)$. That is, we can have $y_{jkilt} > 0$ for these sensors. Constraints (3.16) make use of the flows among the sensors to assign sinks to the sensors. Taking these constraints into account, since $y_{jkilt} > 0$ is possible we set $u_{vjkt} = u_{vilt}$ for all v. For a sink $(i,0)$ at $setlevel = 0$, sink assignment is $u_{ii0t} = 1$ and $u_{vi0t} = 0$ for all $v \neq i$. Then, for a sensor (j,k) at $setlevel = 1$ if $b_{jki} = 1$ we make the sink assignment as $u_{vjkt} = u_{vi0t}$ for all v. Hence, u_{vjkt} is one for only $v = i$, which is the unique sink assignment of the sensor (j,k) in period t. Notice that there can be more than one sink $(i,0)$ with which sensor (j,k) can communicate directly. In this case, only one sink is chosen arbitrarily to make the sink assignment. For the sensors at $setlevel > 1$, the approach is similar. By this way, we satisfy Constraints (3.16) after the determination of flows in the network which will be explained in the following paragraphs.

1. Initialization: Set $\zeta = \left(\sum_{j \in \boldsymbol{N}} \sum_{k \in \boldsymbol{K}} z_{jkt}\right) (\max_{jk} h_{jk})$ and calculate CR^1_{jk} values.
2. While $\{(j,k) : x_{jk} = 1, z_{jkt} = 0, E^{rem}_{jkt} \geq (e^s_k + \zeta(e^r_k + e^c_k))\} \neq \varnothing$ and $CR^1_{jk} > 0$ Do
   2.1. Find a sensor satisfying ($\exists$ a sink $(i,0)$ such that $\exists$ a path from sensor (j,k) to the sink) with the highest CR^1_{jk} value, say sensor (j^*,k^*)
   2.2. Activate sensor (j^*,k^*) and assign it to sink $(i,0)$
   2.3. For all active sensors (j,k) that have no sink assignment Do
      If a sensor (j,k) satisfies ($\exists$ a sink $(i,0)$ such that $\exists$ a path from sensor (j,k) to the sink) Then $u_{ijkt} = 1$
      End For
   2.4. If the network is connected Then Stop
      Else Go to Step 3
      End If
   End While
3. Calculate CR^2_{jk} values
4. While $\{(j,k) : x_{jk} = 0, \exists$ a sensor (i,l) that has a sink assignment or a sink $(i,0)$ with $b_{jki} = 1\} \neq \varnothing$ and $CR^2_{jk} > 0$ Do
   4.1. Find sensor, say (j^*,k^*), with the highest CR^2_{jk} value
   4.2. If $B^{rem} \geq c_{j^*,k^*}$ Then
      4.2.1. Deploy and activate sensor (j^*,k^*) and assign it to sink $(i,0)$
      4.2.2. Repeat Step (2.3)
      4.2.3. If the network is connected Then Stop
         Else $\hat{L} = t - 1$ Stop
         End If
   4.3. Else
      Remove standby sensors in periods $[0,t]$ starting with the ones having the largest cost until enough budget is obtained.
      If $B^{rem} \geq c_{j^*,k^*}$ Then Repeat Steps (4.2.1) - (4.2.3)
      Else $\hat{L} = t - 1$ Stop
      End If
   End If
End While

Figure 4.7. Providing feasibility in period t subject to connectivity restrictions

Proposition 4.6. *The computational complexity of the algorithm in Figure (4.6) is given as* $\mathcal{O}(N^3K^3T + N^5K^4)$.

Proof. Initializing the algorithm is $\mathcal{O}(NK)$. Step (2.1) is $\mathcal{O}(N^2K)$ and step (2.2) is $\mathcal{O}(N^3K^2)$. Then, step 2 is $\mathcal{O}(N^4K^3)$. Satisfiying connectivity restrictions is $\mathcal{O}(N^2K^2T+ N^4K^3)$. Then, step 3 is $\mathcal{O}(N^3K^3T+N^5K^4)$. As a result, providing feasibility in period t subject to sink assignment constraints is $\mathcal{O}(N^3K^3T + N^5K^4)$. □

One important point is that after the set generation algorithm explained above we can have sensors that do not belong to any of these sets. This means these sensors cannot communicate any of the sensors in the sets. Hence, we cannot assign sinks to these sensors with our algorithm. At this point, we use the algorithm given in Figure (4.7) to maintain the communication in the network. We first consider to activate standby sensors since the strategy is free. A standby sensor to activate is selected in a greedy way among the ones that have sufficient remaining energy and can communicate directly with at least one of the sensors in the sets and at least one of the sensors that are outside the sets. We determine a communication ratio for each standby sensor (j,k) in period t as

$$CR_{jk}^{1^{(r)}} = \frac{\left(\sum_{(i,l)\in\{(i,l)|u_{vilt}=0\ \forall v\}} a_{ijk}\right) x_{jk}^{(r)} E_{jkt}^{rem}}{c_{jk}} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K} \quad (4.50)$$

where E_{jkt}^{rem} represents the remaining energy of a sensor (j,k) in period t.

Then, we activate the standby sensors starting from the one with the highest positive $CR_{jk}^{1^{(r)}}$ value and continue until all sensors find a set or we do not have standby sensors with positive $CR_{jk}^{1^{(r)}}$ value. Observe that, by activating a standby sensor we can insert more than one isolated sensor in a set, since either isolated sensors can communicate with each other or the activated sensor is in the communication range of more than one isolated sensor. After the activation of a standby sensor, both possibilities are checked in order to provide communication with the least number of additional sensors. If there are still sensors outside the sets then we consider to deploy new sensors and

activate them, i.e. $x_{jk} = z_{jkt} = 1$, in period t. In order to determine which sensors to deploy and activate in a period t, we calculate another communication ratio, i.e. $CR_{jk}^{2(r)}$, for each sensor (j,k) as:

$$CR_{jk}^{2(r)} = \frac{\left(\sum_{(i,l)\in\{(i,l)|u_{vilt}=0\ \forall v\}} a_{ijk}\right)\left(1 - x_{jk}^{(r)}\right) E_k}{c_{jk}} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K} \quad (4.51)$$

where E_k represents the initial battery energy of a type-k sensor.

It is possible that the remaining budget is not adequate to deploy the candidate sensor that has largest $CR_{jk}^{2(r)}$ value. In this case, we delete unused sensors until the current period to find the necessary fund. If we cannot obtain sufficient budget to deploy the selected sensor then we update $\hat{L} = t - 1$ and stop the algorithm. We continue with the algorithm until we can find a sensor that has positive $CR_{jk}^{2(r)}$ value and we can deploy the sensor either with the remaining budget or the generated budget after removing some of the sensors.

Proposition 4.7. *The computational complexity of the algorithm in Figure (4.7) is given as* $\mathcal{O}(N^2K^2T + N^4K^3)$.

Proof. Calculating ζ is $\mathcal{O}(NK)$ and calculating CR_{jk}^1 values is $\mathcal{O}(N^2K^2)$. This means, initializing the algorithm is $\mathcal{O}(N^2K^2)$. Finding a sensor (j^*, k^*) according to the conditions in step (2.1) is $\mathcal{O}(N^2K)$. Assigning sink for active sensors in step (2.3) is $\mathcal{O}(N^3K^2)$. Checking connectivity is $\mathcal{O}(NK)$. Therefore, step 2 is $\mathcal{O}(N^4K^3)$. Calculating CR_{jk}^2 values is $\mathcal{O}(N^2K^2)$ and finding the maximum one is $\mathcal{O}(NK)$. Step (4.2) is $\mathcal{O}(N^3K^2)$ and step (4.3) is $\mathcal{O}(NKT+N^3K^2)$. Therefore, step 4 is $\mathcal{O}(N^2K^2T+N^4K^3)$. As a result, providing feasibility in period t subject to connectivity restrictions is $\mathcal{O}(N^2K^2T + N^4K^3)$. □

The last step of the greedy algorithm aims to find the values of the data flows in the network. For this purpose, for each period t within the lifetime $\hat{L}$ we consider the Routing Problem (RP), which tries to find the minimum energy consuming sensor-to-

sink data flow paths.

RP :

$$\min \quad e_k^r \sum_{i\in \boldsymbol{N}} \sum_{l\in \boldsymbol{K}} y_{iljk} + e_k^c \sum_{i\in \boldsymbol{N}} \sum_{l\in \boldsymbol{K'}} y_{jkil} \tag{4.52}$$

s.t.

$$\sum_{i\in \boldsymbol{N}} \sum_{l\in \boldsymbol{K}} y_{iljk} + h_{jk}\bar{z}_{jkt} = \sum_{i\in \boldsymbol{N}} \sum_{l\in \boldsymbol{K'}} y_{jkil} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K} \tag{4.53}$$

$$\sum_{j\in \boldsymbol{N}} \sum_{k\in \boldsymbol{K}} y_{jki0} = \sum_{j\in \boldsymbol{N}} \sum_{k\in \boldsymbol{K}} h_{jk}\bar{u}_{ijkt} \qquad i \in \boldsymbol{N} \tag{4.54}$$

$$\sum_{i\in \boldsymbol{N}} \sum_{l\in \boldsymbol{K'}} y_{jkil} \leq M_1 \bar{z}_{jkt} \qquad j \in \boldsymbol{N};\ k \in \boldsymbol{K} \tag{4.55}$$

$$\sum_{i\in \boldsymbol{N}} \sum_{l\in \boldsymbol{K}} y_{ilj0} \leq M_2 x_{j0} \qquad j \in \boldsymbol{N} \tag{4.56}$$

$$y_{iljk} \leq M_2 \left(2 - \sum_{v\in \boldsymbol{N}} |\bar{u}_{vilt} - \bar{u}_{vjkt}|\right)/2 \qquad i,j \in \boldsymbol{N};\ k \in \boldsymbol{K'};\ l \in \boldsymbol{K} \tag{4.57}$$

$$y_{iljk} \leq M_2 b_{ilj} \qquad i,j \in \boldsymbol{N};\ k \in \boldsymbol{K'};\ l \in \boldsymbol{K} \tag{4.58}$$

$$y_{iljk} \geq 0, \tag{4.59}$$

where $\{\bar{\mathbf{L}}^{(\mathbf{r})}, \bar{\mathbf{n}}^{(\mathbf{r})}, \bar{\mathbf{x}}^{(\mathbf{r})}, \bar{\mathbf{z}}^{(\mathbf{r})}, (\bar{\mathbf{g}}^{(\mathbf{r})}), \mathbf{y}^{(\mathbf{r})}, \mathbf{w}^{(\mathbf{r})}\}$ is the solution after applying the first three steps of the greedy heuristic GH.

Actually, at this step of the algorithm we should decide on the values for both data flows $\mathbf{y}$ and dummy variables $\mathbf{w}$. Notice that variables $\mathbf{y}$ are continuous and variables $\mathbf{w}$ are binary. Therefore, if we have included the variables $\mathbf{w}$ to the RP, then we would have a MILP instead of a LP, which is obviously more difficult to solve. Then, we add constraints (4.57) to satisfy constraints (3.16). Constraints (4.57) calculate the absolute difference among the sink assignment variables of sensors (i,l) and (j,k). Observe that since each sensor is assigned to only one sink, total difference can be either zero means they are assigned to same sink or two means they are assigned to different sinks. According to constraints (3.16), if there is a positive flow from sensor (i,l) to (j,k), then the sink assignments can be the same and they have to be the same

as we have shown in Proposition 3.3. Constraints (4.57), allowing a possible flow from sensor (i,l) to sensor (j,k) only if they have the same sink assignments, guarantee the feasibility with respect to Constraints (3.16). Therefore, setting $w_{iljkt} = 1$ if $y^*_{iljk} > 0$ and zero otherwise, where y^* is an optimal flow obtained by solving RP at period t, is feasible with respect to constraints (3.20) and (3.21).

Proposition 4.8. *Based on the Karmarkar's interior point algorithm (Karmarkar, 1984), the solution algorithm for RP is of complexity $\mathcal{O}(N^6K^6L_B)$ where L_B is the size of the LP instance in terms of the number of bits necessary for storage.*

Proof. According to the potential reduction algorithm developed by Ye (1991), Karmarkar's interior point algorithm can be solved in $\mathcal{O}(n^3L)$ number of iterations where n is the number of variables and L number of bits required for storage. For our case, number of variables of the RP is $\mathcal{O}(N^2K^2)$. Number of constraints is $\mathcal{O}(N^2K^2)$. □

Then combining the subalgorithms described above, we can reach the following proposition related with the overall cost of the greedy algorithm.

Proposition 4.9. *The computational complexity of the algorithm in Figure (4.3) is given as $\mathcal{O}(N^2K^2T^2 + \tau_1N^3KT + \tau_1N^2KT^2 + N^3K^3T + N^5K^4 + N^6K^6L_B)$ where $\tau_1 = \max_i \left\{\sum_{j\in\boldsymbol{N}}\sum_{k\in\boldsymbol{K}} a_{ijk}\right\}$ is the maximum number of points in the sensor field that a sensor can cover and L_B is the size of the RP instance in terms of the number of bits necessary for storage.*

Proof. Calculating total energy consumption of a sensor (j,k) upto period t is $\mathcal{O}(NKT)$. Calculating step (1.1) is $\mathcal{O}(NKT^2)$. Calculating step (1.2) is $\mathcal{O}(T)$. Then, step 1 is $\mathcal{O}(N^2K^2T^2)$. Complexity of step 2 is $\mathcal{O}(\tau_1N^3KT + \tau_1N^2KT^2)$. Complexity of step 3 is $\mathcal{O}(N^3K^3T + N^5K^4)$. Step 4 is $\mathcal{O}(N^6K^6L_B)$. As a result, greedy heuristic is $\mathcal{O}(N^2K^2T^2 + \tau_1N^3KT + \tau_1N^2KT^2 + N^3K^3T + N^5K^4 + N^6K^6L_B)$. □

4.4.2. Discrimination Heuristic

Discrimination heuristic (DH) is another approach to generate a feasible solution from the solution of the Lagrangean subproblem at iteration r. Actually, this heuristic uses almost the same subalgorithms with greedy heuristic but in different order. Greedy heuristic first tries to restore the feasibility with respect to the coverage and budget constraints in all periods within the planning horizon. After this, the connectivity is checked for all periods within the lifetime while making the sink assignments. Finally, for all periods within the lifetime data flow routes are determined. One major drawback of this procedure is that we are consuming most of our budget while we are trying to satisfy coverage constraints for more periods as possible. This decreases our chance to provide connectivity by deploying new sensors. Hence we can end up with unsatisfactory lower bound. On the other hand, discrimination heuristic considers feasibility with respect to coverage, budget and connectivity constraints and the determination of data routes for each period independently. This means, we cannot move to the next period without satisfying all relaxed constraints in a period.

We use the algorithm given in Figure (4.8) to obtain a feasible solution to PSRPC. For period t we check the coverage and budget constraints. If the coverage constraints are satisfied with the available budget, then we use algorithm in Figure (4.6) to assign sinks to the active sensors. We consider to remove sensors that are in standby mode until period t when the budget is over. If connectivity is achieved, using the algorithm in Figure (4.9) we eliminate active sensors whose removal does not affect the coverage and connectivity of the network in period t starting from the most expensive one.

Proposition 4.10. *The computational complexity of the algorithm in Figure (4.9) is given as* $\mathcal{O}(N^4K^3)$.

Proof. Reassigning the sinks to the sensors is $\mathcal{O}(N^3K^2)$. This process is repeated for NK times. Hence, eliminating unnecessary sensors is $\mathcal{O}(N^4K^3)$. □

1. Apply Step 1 of the algorithm in Figure (4.3).
2. Set $L_{LB}^{(r)} = T$
3. For all $t \leq L_{LB}^{(r)}$ Do

    If every point is covered Then

        If budget constraint is satisfied Then

        3.1. Use the algorithm in Figure (4.6) to generate a feasible solution subject to the sink assignment constraints.

        3.2. If the network is connected Then

        $t \leftarrow t + 1$

        3.2.1 Use the algorithm in Figure (4.9) to eliminate the active sensors that do not harm coverage or connectivity restrictions.

        3.2.2. Solve RP to determine data flows $\mathbf{y}$. Set $L_{LB} = \bar{L}$ and Stop

        Else $\bar{L} = t - 1$ and Stop

        End If

    Else

        Remove standby sensors in periods $[0, t]$ starting with the ones having the largest cost until budget constraint is satisfied.

        If this is not possible Then $\bar{L} = t - 1$ and Stop

        Else Repeat Steps (3.1) and (3.2)

        End If

    End If

Else

    Apply Step (2.2) of the algorithm in Figure (4.5)

    If coverage constraints are satisfied Then Repeat Steps (3.1) and (3.2)

    Else $\bar{L} = t - 1$ and Stop

End If

End For

Figure 4.8. Discrimination heuristic, DH

1. While $\{(j,k) : z_{jkt} = 1$ and $\forall v$ with $a_{vjk} = 1$, $\exists$ sensor (i,l) $z_{ilt} = 1$ with $a_{vil} = 1\} \neq \varnothing$ Do
   1.1. While $\{(j,k) : z_{jkt} = 1$ and $\forall v$ with $b_{jkv} = 1$, $\exists$ sensor (i,l) $z_{ilt} = 1$ with $b_{ilv} = 1\} \neq \varnothing$ Do
      $z_{jkt} = 0$
      Reassign the sinks to the remaining active sensors
   End While
End While

Figure 4.9. Eliminating unnecessary sensors from network in period t

Data flow routes are determined with the linear program RP as explained in the previous subsection and move to the next period. In case the coverage is not satisfied in period t, we use algorithms given in Figure (4.4) and Figure (4.5). If we can satisfy the coverage constraints, then we continue with finding the sink assignments and the corresponding data flow routes, in period t.

Proposition 4.11. *The computational complexity of the algorithm in Figure (4.8) is given as* $\mathcal{O}(N^3K^3T^2 + N^5K^4T + N^4K^3T + N^6K^6TL_B + \tau_1 N^3KT + \tau_1 N^2KT^2)$ *where* $\tau_1 = \max_i \left\{\sum_{j\in\boldsymbol{N}}\sum_{k\in\boldsymbol{K}} a_{ijk}\right\}$ *is the maximum number of points in the sensor field that a sensor can cover and* L_B *is the size of the RP instance in terms of the number of bits necessary for storage.*

Proof. The complexity of step 1 is $\mathcal{O}(N^2K^2T^2)$. Checking coverage is of $\mathcal{O}(N^2K)$ and checking budget takes $\mathcal{O}(NK)$. Satisfying sink assignment constraints is $\mathcal{O}(N^3K^3T + N^5K^4)$. Checking connectivity is $\mathcal{O}(NK)$. Eliminating unnecessary sensors from the network is $\mathcal{O}(N^4K^3)$. Solving LP is $\mathcal{O}(N^6K^6L_B)$. Removing standby sensors from the network is $\mathcal{O}(NKT)$. Applying step (2.2) to provide feasibility subject to coverage and budget constraints is $\mathcal{O}(\tau_1 N^3K + \tau_1 N^2KT)$. Then, the complexity of the algorithm is $\mathcal{O}(N^3K^3T^2 + N^5K^4T + N^4K^3T + N^6K^6TL_B + \tau_1 N^3KT + \tau_1 N^2KT^2)$. □

5. THE SINK LOCATION, SENSOR PLACEMENT, SCHEDULING AND ROUTING PROBLEM WITH CONNECTIVITY RESTRICTIONS

5.1. Introduction

We have developed solution strategies for PSRPC which assumes that we are given the sink locations in the previous chapter. These solution techniques can be utilized to find good, possibly optimal, sink locations. In this chapter given the number of sinks in the sensor field, we aim to develop heuristics to find good locations for sinks in order to maximize the network lifetime.

5.2. Model Formulation

The mathematical model that locates the sinks, places sensors, determines the activity schedules of the sensors and constructs the sensor-to-sink data flow routes under connectivity restrictions (LPSRPC) can be given as follows:

$LPSRPC$:

$$\max \quad L \tag{5.1}$$

$$\text{s.t.} \quad (3.2)-(3.15),(3.17)-(3.19),(3.20),(3.22),(3.23),(3.25)$$

$$\sum_{i \in \boldsymbol{N}} x_{i0} = S \tag{5.2}$$

where S is the given number of sinks in the WSN.

The median constraint (5.2) guarantee to place S sinks in the network. One important point related with the sinks is that they are special types of sensors, hence deploying a sink on a point i is costless and sink $(i, 0)$ will be in active mode during the lifetime of the network without any energy constraint.

One can observe that when the locations of S sinks are given, i.e. x_{i0} values are known, the problem will reduce to PSRPC for which we have introduced solution procedures in the previous section. Since we have finite number of points to locate the sinks, we can find the best locations for sinks by simply enumarating over all possible combinations. However, as the number of candidate locations increases linearly, the number of possible combinations increase exponentially. Therefore, instead of trying all alternative locations for S sinks, we can develop search algorithms to find good locations for sinks that maximizes the lifetime L of the network. Two different search algorithms are built: a local search heuristic and tabu search heuristic whose details are explained in the following section.

5.3. Solution Procedures

5.3.1. Local Search Heuristic

Local search techniques are frequently used for solving hard combinatorial optimization problems such as the p-median problem (Arya et al., 2004). We make use of similar searching strategy in order to locate S sinks. A solution $\mathcal{S} = \{s_1, ..., s_S\}$ will be the set of locations of S sinks and $\mathcal{B}_{\mathcal{N}_s}(\mathcal{S})$ will be the set of all neighbors around the solution $\mathcal{S}$ with the neighborhood function $\mathcal{N}_s$. In order to generate neighbors around the current solution $\mathcal{S}$, we swap the locations of randomly selected s sinks out of S sinks of the current solution. Therefore, for our problem $\mathcal{N}_s$ will be s-swap neighborhood function. Observe that, s can take different values, i.e. $s = 1, ..., S$, which gives alternative neighborhood functions. Moreover, we can calculate the size of the neighborhood $\mathcal{B}_{\mathcal{N}_s}(\mathcal{S})$, i.e. the number of solutions in $\mathcal{B}_{\mathcal{N}_s}(\mathcal{S})$, when we are given the neighborhood function $\mathcal{N}_s$. For a $\mathcal{N}_s$ neighborhood function, the size of the neighborhood NS_s can be calculated as

$$NS_s = \binom{N-S}{s}\binom{S}{s} \qquad s \in \{1, ..., S\} \tag{5.3}$$

where N is the number of candidate locations to locate a sink.

The local search heuristic is given in Figure (5.1) and starts with an initial solution $\mathcal{S}_0$, which is obtained by locating S sinks randomly. An initial network lifetime L_{LB} is calculated using the sink locations $\mathcal{S}_0$ with one of the feasible solution generation algorithm, namely greedy or discriminative heuristic. For a given solution $\mathcal{S}$ and for each s-swap neighborhood of $\mathcal{S}$, we consider the P_s percentage of the neighborhood to search an improving solution. Then, we select the best improving solution among all s-swap neighborhoods, $s = 1, ..., S$, to update the network lifetime L_{LB}. We continue with improving the lower bound for the network lifetime L until the algorithm run for *iterlim* many iterations or we cannot update the current L_{LB} for NI consecutive iterations.

Notice that, if we are searching for all solutions in the s-swap neighborhoods, i.e. $P_s = 1$, and we cannot update the current lower bound at last iteration, then we can stop since we are at a local optimum. This means, if we decide to take $P_s = 1$, then we can select $NI = 1$ since choosing NI greater than one is meaningless. On the other hand, picking a $P_s \in (0, 1)$ decreases the power of the algorithm in terms of intensification in a neighborhood. As a result, one can choose larger NI values while decreasing the P_s value not to lose much from the power of the algorithm.

Proposition 5.1. *The computational complexity of the solution procedure for LS is* $\mathcal{O}(\tau_2 S^3(iterlim) + \tau_2 O_{LB} S(iterlim))$ *where* $\tau_2 = \max_s \{NS_s P_s\}$ *is the maximum number of solutions generated from a s-swap neighborhood and* $\mathcal{O}(O_{LB})$ *represents the complexity of the algorithm to generate a lower bound.*

1. Initialization: Locate S sinks randomly, use the algorithm GH (or DH) to generate an initial lower bound, L_{LB}. Set $t = 0$, $noimpr = 0$ and $L^*_{LB} = 0$.
2. While $t < iterlim,\ noimpr \leq NI$ Do

    For all $s \leq S$ Do

        count $= 0$, $L_{LB} = 0$;

        While $count \leq NS_sP_s$ Do

            Change locations of s-sinks randomly and use the algorithm GH (or DH) to generate a lower bound, $\bar{L}$.

            Update $L_{LB} = \max\left\{\bar{L}, L_{LB}\right\}$ and $count \leftarrow count + 1$

        End While

    End For

    Update $L^*_{LB} = \max\left\{L_{LB}, L^*_{LB}\right\}$ and set $noimpr = 0$.

    If L^*_{LB} is not updated Then

        $noimpr \leftarrow noimpr + 1$

    End If

    $t \leftarrow t + 1$

End While

Figure 5.1. Local search heuristic, LS

Proof. Expected number of iterations to locate S sinks is $\mathcal{O}(S^2)$. Finding lower bound is $\mathcal{O}(O_{LB})$. Then, initializing the algorithm is $\mathcal{O}(S^2 + O_{LB})$. Changing the locations of s-sinks is $\mathcal{O}(S^2)$. Finding best lower bound in the s-swap neighborhood is $\mathcal{O}(\tau_2 S^2 + \tau_2 O_{LB})$. Doing this for all s-swap neigborhoods is $\mathcal{O}(\tau_2 S^3 + \tau_2 O_{LB} S)$. We repeat this until the stopping condition is satisfied. Hence, complexity of step 2 is $\mathcal{O}(\tau_2 S^3(iterlim) + \tau_2 O_{LB} S(iterlim))$ which is also the complexity of the LS algorithm. □

5.3.2. Tabu Search Heuristic

Second search heuristic for the sink locations is a tabu search algorithm. Different than the local search algorithm, tabu search aims to visit as distict parts of the solution space as possible by forbidding to revisit the recent *tabutenure* many solutions from the solution space of sink locations (Gendreau and Potvin, 2005). The superiority of tabu search algorithm over the local search is that tabu search can both search a neighborhood intensively and search diverse regions of the solution space at the same time whereas local search algorithm even though performing well in intensification, can be very poor in diversification.

As in the local search, we are using S different neighborhood functions $\mathcal{N}_s$, namely s-swap neighborhood functions for $s = 1, ..., S$. The tabu search algorithm described in Figure (5.2) locates S sinks initially. This solution is used to find an initial lower bound L_{LB} for the network lifetime and added to the *tabulist* in order to prohibit returning to this solution for at least *tabutenure* many iterations. We search for an improving solution in the P_s percentage of the neighborhood $\mathcal{B}_{\mathcal{N}_s}(\mathcal{S})$ of a current solution $\mathcal{S}$ for all s. The L_{LB} is updated with the best improving lower bound for network lifetime L and the corresponding sink locations, which will be in the *tabulist* for the next *tabutenure* updates, will be the new locations for sinks. The algorithm stops after *iterlim* many iterations or NI many consecutive nonimproving iterations.

Notice that, both local search and tabu search algorithms move to the steepest ascent neighbor from the current solution. One may prefer to move to an improving or even a nonimproving solution with a probability which can be helpful to reach other parts of the solution space or we can allow a move although it is tabu if there is no risk of cycling to deeply search a particular region. The aspirations in the algorithms may result in a better lower bound for network lifetime L (Sacchi and Armentano, 2010).

Proposition 5.2. *The computational complexity of the solution procedure for TS is* $\mathcal{O}(\tau_2 S^3(tabutenure)(iterlim) + \tau_2 O_{LB} S(iterlim))$ *where* $\tau_2 = \max_s \{NS_s P_s\}$ *is the maximum number of solutions generated from a s-swap neighborhood and* $\mathcal{O}(O_{LB})$ *represents the complexity of the algorithm to generate a lower bound.*

Proof. Initializing the algorithm is $\mathcal{O}(S^2 + O_{LB})$. Finding non-tabu locations for s-sinks is $\mathcal{O}(S^2(tabutenure))$. Finding best lower bound in the s-swap neighborhood is $\mathcal{O}(\tau_2 S^2(tabutenure) + \tau_2 O_{LB})$. Doing this for all s-swap neigborhoods is $\mathcal{O}(\tau_2 S^3(tabutenure) + \tau_2 O_{LB} S)$. Adding a sink set $\{s_1, ..., s_S\}$ to the tabu list is $\mathcal{O}(S(tabutenure))$. We repeat these until the stopping condition is satisfied. Hence, complexity of step 2 is $\mathcal{O}(\tau_2 S^3(tabutenure)(iterlim) + \tau_2 O_{LB} S(iterlim))$ which is also the complexity of the TS algorithm. □

1. Initialization: Locate S sinks randomly, add $\{s_1,\ ...,\ s_S\}$ to *tabulist* and use the algorithm GH (or DH) to generate an initial lower bound, L_{LB}. Set $t = 0$, $noimpr = 0$ and $L^*_{LB} = 0$.
2. While $t < iterlim,\ noimpr \leq NI$ Do

    For all $s \leq S$ Do

        count $= 0$, $L_{LB} = 0$;

        While $count \leq NS_sP_s$ Do

            Change locations of s-sinks randomly until we find a sink set $\{s_1,\ ...,\ s_S\} \notin tabulist$.

            Use the algorithm GH (or DH) to generate a lower bound, $\bar{L}$.

            Update $L_{LB} = \max\left\{\bar{L}, L_{LB}\right\}$ and $count \leftarrow count + 1$

        End While

    End For

    Update $L^*_{LB} = \max\{L_{LB}, L^*_{LB}\}$, add $\{s^*_1,\ ...,\ s^*_S\}$ to *tabulist* and set $noimpr = 0$.

    If L^*_{LB} is not updated Then

        $noimpr \leftarrow noimpr + 1$

    End If

    $t \leftarrow t + 1$

End While

Figure 5.2. Tabu search heuristic, TS

6. COMPUTATIONAL RESULTS

6.1. Introduction

In this chapter we evaluate experimentally the performance of the developed methods in the thesis. We first evaluate the performance of the two Lagrangean Heuristics for PSRPC and find the accuracy of the solution methods. Then, we test the solutions found by search algorithms using Greedy Heuristic and Discrimination Heuristic. Finally, we evaluate the sensitivity of the algorithms to the number of sinks.

6.2. Test Environment

All the experiments are carried out on a computer with Intel Xeon 3.40 GHz processor and 32 GB of RAM working under Windows 2003 Server operating system. In the experiments we assume that the sensor field has a square $n \times n$ grid structure with n^2 points and the coverage requirements are selected as $f_i = 2$ for all points in the sensor field. There are $K = 2$ sensor types, and parameter values are depend on the sensor type given in Table 6.1. The value of e_k^s is based on a period length 12 hours. We are assuming that every half an hour, i.e. 1800 seconds, a data packet is generated which is equivalent to 24 data packets in a period, i.e. $h_{jk} = 24$ data packets/period. The values for the parameters e_k^s, e_k^r, e_k^c and E_k are based on the experimental results for a Mica2 mote studied by Torres and Kabara (2006). In the experiments, for each

Table 6.1. Sensor specifications

k	c_{jk}	r_k^s	r_k^c	e_k^s (Joules)	e_k^r (Joules)	e_k^c (Joules)
1	rand (1, 10)	1	1.5	744	0.01	0.013
2	rand (c_{j1}, c_{j1} + 5)	2	3	744	0.01	0.018

network instance we assign three budget B levels, namely low, medium and high under three initial battery energy E_k levels again low, medium and high. The performance of the search heuristics are tested with two values of the number of sinks, S.

6.3. Results for Sensor Placement, Scheduling and Routing Problem with Connectivity Restrictions

In this section we evaluate the accuracy and efficiency of the First Lagrangean Heuristic (LH_1) and Second Lagrangean Heuristic (LH_2) developed in Chapter 3. They are both generating feasible solutions with Greedy Heuristic (GH). Energy levels used in these experiments are summarized in Table 6.2 and the formulas to calculate the budget levels are given in Table 6.3. In all instances we aim to cover each point in the sensor field by at least one active sensor in each period over planning horizon $T = 30$. We compare the results provided by LH_1 and LH_2 with those given by the solver CPLEX 11.0 (ilog, 2007).

Table 6.2. Energy levels used in PSRPC runs

k	E_{low}	E_{medium}	E_{high}
1	1000	2000	3000
2	2000	3000	4000

The initial battery energies of the sensors given in Table 6.2 are lower than the real ones. As we have explained in Chapter 3, both of the algorithms LH_1 and LH_2 are exponential, hence we cannot reach any solution with these algorithms under real energy parameters. Therefore, we evaluate the performances of the Lagrangean heuristics on simplified instances and use more accurate parameters in the runs for search algorithms which are polynomial. Similarly, budgets are lower than the ones in the experiments for search algorithms.

Table 6.3. Formulas for the budget levels used in PSRPC runs

B_{low}	$\sum_{j \in \boldsymbol{N}} (0.75\ c_{j1} + 0.25\ c_{j2})\ /\ 4$
B_{medium}	$\sum_{j \in \boldsymbol{N}} (0.50\ c_{j1} + 0.50\ c_{j2})\ /\ 3$
B_{high}	$\sum_{j \in \boldsymbol{N}} (0.25\ c_{j1} + 0.75\ c_{j2})\ /\ 2$

The results for two Lagrangean Heuristics, summarized in Table 6.4, include the lower bounds obtained with Greedy Heuristic and the upper bounds calculated from the respective relaxed model on the optimal value of the network lifetime. We denote by

L^*_{LB} the best lifetime value, which is a lower bound on the optimal objective, generated by GH over the iterations of LH_1 and LH_2. We represent the best upper bound that is found in all iterations of LH_1 and LH_2 by L^*_{UB}. In order to generate the optimal solutions, we use CPLEX 11.0 with the default options. In each problem instance a time limit of three hours, which is larger than the CPU time of the Lagrangean heuristics for the same instance, is given to CPLEX 11.0. The imposed time limit is either sufficient to find an optimum solution or a feasible solution, or CPLEX 11.0 cannot reach any solution. An optimum solution for the problem instance is indicated with an (*) in the table under the L^{IP}_{LB} column. If CPLEX 11.0 stops with a feasible solution than this will be a lower bound on the optimum objective value and given in column L^C_{LB}. If there is no solution found by CPLEX 11.0 at the end of imposed time limit, we report this with a (—) in the table. Since CPLEX 11.0 cannot give a feasible solution for all instances of 5 × 5 grid network, we do not continue with CPLEX 11.0 for larger problem instances. For the upper bound, the linear relaxation of a problem instance is solved and listed under the L^{LP}_{UB} column. We also set some limits on both of the Lagrangean heuristics, since they include MILP models to be solved at each iteration. The heuristics LH_1 and LH_2 run for at most $iterlim = 1000$ iterations while the value of π, which is initially two and halved after each 15 consecutive nonimproving iterations for the best lower bound, is greater than 0.05 and the difference between the best upper bound and best lower bound is greater than % 10 of the best upper bound. We solve SP_3 of LH_1 until the gap between the best lower and upper bounds is less than ‰ 5. In addition to this limitation, we run SP_2 of LH_2 at most six hours since SP_2 of LH_2 involves additional constraints, which makes the problem more complicated, besides the constraints of SP_3 in LH_1. The CPU times in seconds are listed for the algorithms LH_1 and LH_2 and for CPLEX 11.0 to find a solution for PSRPC.

We see that CPLEX 11.0 can find an optimal solution only for 4 × 4 grid instance with low energy and low budget levels, gives a feasible solution for six instances in 4 × 4 grid and cannot give any solution for the rest of the instances. These results show how it is difficult to generate an optimum solution even for the small instances of PSRPC. The instance that is optimally solved by CPLEX 11.0 is also solved by LH_1 and LH_2 to optimality. For the six instances that CPLEX 11.0 gives a feasible

solution, the algorithm LH_1 finds better solutions than CPLEX 11.0 for two of them and gives the same results for the others whereas the algorithm LH_2 finds better lower bounds for three instances and worse lower bounds for two instances. Comparing the Lagrangean Heuristics for the 15 instances among the ones that we can solve with LH_1 and LH_2, first heuristic gives better lower bounds and for one instance LH_1 gives worse lower bound. In 26 of the instances LH_1 finds better upper bounds than LH_2. We observe that as the size of the problem instance increases, the performance of LH_1 in finding upper bound decreases. This can be due to the requirement of more iterations to converge to a good upper bound when the problem gets larger. We observe that the average CPU time of LH_1 is larger than the one of LH_2 for all energy levels as shown in Figure 6.1 - Figure 6.3. This can be the result of our termination criteria used in SP_3 of LH_1 and SP_2 of LH_2. As the size of the instance gets larger, to solve SP_3 until the criterion is satisfied becomes more time consuming at each iteration which requires more time to complete the given number of iterations. Another reason can be since the best upper bound found by the algorithm LH_2 is not updated in all iterations, it is possible that the best lower bound is also not updated for the most of the iterations which quickly decreases the value of π below the critical value hence the algorithm terminates before the Lagrangean heuristic LH_1.

We express the accuracy of the methods as percent deviation from the best known upper bound. For instance the accuracy of LH_1 can be given as

$$100 \times \frac{\min(L_{UB}^{LP}, L_{UB}^{*}) - L_{LB}^{*}}{\min(L_{UB}^{LP}, L_{UB}^{*})} \ . \tag{6.1}$$

An accurate solution method is expected to have small accuracy calculated as in above formula since closer best lower and upper bounds means we have more information about where the optimum solution lies. From this point of view, we observe that for all energy levels LH_1 has the lowest average accuracy value whereas LH_2 has smaller average accuracy level than CPLEX11.0. Hence, we can conclude that on average solution method LH_1 is more accurate than LH_2 with feasible solution generation algorithm GH and both of them are performing better than CPLEX11.0.

Table 6.4. Comparison of LH_1 and LH_2 with CPLEX 11.0 at low energy level

		Network Lifetime						Accuracy			CPU Time (s)		
		LH_1		LH_2		CPLEX11.0							
n × n	B	L^*_{LB}	L^*_{UB}	L^*_{LB}	L^*_{UB}	L^{IP}_{LB}	L^{LP}_{UB}	LH_1	LH_2	CPLEX11.0	LH_1	LH_2	CPLEX11.0
	15	2	2.91	2	30	2*	30	31.24	93.33	93.33	9883.787	6744.597	2250.14
4 × 4	30	3	3.28	3	30	3	30	8.50	90.00	90.00	2990.06	6624.62	10800
	59	5	8.56	5	30	—	30	41.59	83.33	—	3682.13	7520.10	10800
	27	2	30	2	30	—	30	93.33	93.33	—	46009.50	18274.23	10800
5 × 5	55	4	4.44	3	30	—	30	9.90	90.00	—	12904.26	16133.41	10800
	104	6	8.50	5	30	—	30	29.38	83.33	—	22500.05	18430.91	10800
	39	2	30	2	30	—	—	93.33	93.33	—	3202442.35	33084.26	—
6 × 6	77	4	6.09	3	30	—	—	34.34	90.00	—	111620.50	32306.56	—
	144	7	10.77	5	30	—	—	35.03	83.33	—	103420.80	31847.24	—
	52	2	30	2	30	—	—	93.33	93.33	—	862492.48	69390.42	—
7 × 7	107	4	30	4	30	—	—	86.67	86.67	—	885941.39	74043.06	—
	198	6	30	6	30	—	—	80.00	80.00	—	123761.86	61524.61	—
	69	—	—	2	30	—	—	—	93.33	—	—	130976.42	—
8 × 8	142	—	—	4	30	—	—	—	86.67	—	—	162231.8	—
	261	—	—	6	30	—	—	—	80.00	—	—	293734.13	—
	85	—	—	2	30	—	—	—	93.33	—	—	86526.10	—
9 × 9	177	—	—	4	30	—	—	—	86.67	—	—	86469.08	—
	328	—	—	7	30	—	—	—	76.67	—	—	87646.00	—
Average:		3.92	14.96	3.72	30	2.50	30	53.05	87.59	91.67	448970.76	67972.64	9375.02

Table 6.5. Comparison of LH_1 and LH_2 with CPLEX 11.0 at medium energy level

		Network Lifetime						Accuracy			CPU Time (s)		
		LH_1		LH_2		CPLEX11.0							
n × n	B	L^*_{LB}	L^*_{UB}	L^*_{LB}	L^*_{UB}	L^{IP}_{LB}	L^{LP}_{UB}	LH_1	LH_2	CPLEX11.0	LH_1	LH_2	CPLEX11.0
	15	4	19.64	2	30	4	30	79.63	93.33	86.67	3682.13	6729.37	10800
4 × 4	30	6	14.28	4	30	6	30	57.98	86.67	80.00	13751.88	6627.92	10800
	59	10	13.73	10	30	1	30	27.14	66.67	96.67	25404.54	7546.91	10800
	27	2	30	0	30	—	30	93.33	100.00	—	283789.32	18266.50	10800
5 × 5	55	6	10.61	6	30	—	30	43.43	80.00	—	101441.66	16020.53	10800
	104	11	17.54	10	30	—	30	37.29	66.67	—	66193.28	18508.25	10800
	39	4	22.36	4	30	—	—	82.11	86.67	—	693095.56	32773.20	—
6 × 6	77	8	11.52	8	30	—	—	30.52	73.33	—	148902.59	30478.73	—
	144	12	20.27	12	30	—	—	40.79	60.00	—	345552.02	32178.80	—
	52	4	30	4	30	—	—	86.67	86.67	—	98946.27	66403.31	—
7 × 7	107	8	30	8	30	—	—	73.33	73.33	—	97380.83	68888.76	—
	198	12	30	12	30	—	—	60.00	60.00	—	89159.32	63061.42	—
	69	4	30	4	30	—	—	86.67	86.67	—	117815.68	141493.68	—
8 × 8	142	8	30	8	30	—	—	73.33	73.33	—	196689.18	140849.80	—
	261	—	—	14	30	—	—	—	53.33	—	—	285759.92	—
	85	—	—	4	30	—	—	—	86.67	—	—	86987.47	—
9 × 9	177	—	—	8	30	—	—	—	73.33	—	—	86480.70	—
	328	—	—	10	30	—	—	—	66.67	—	—	86533.44	—
Average:		7.07	22.14	7.11	30	3.67	30	62.30	76.30	87.78	162986.02	66421.60	10800

Table 6.6. Comparison of LH_1 and LH_2 with CPLEX 11.0 at high energy level

		Network Lifetime						Accuracy			CPU Time (s)		
		LH_1		LH_2		CPLEX11.0							
n × n	B	L^*_{LB}	L^*_{UB}	L^*_{LB}	L^*_{UB}	L^{IP}_{LB}	L^{LP}_{UB}	LH_1	LH_2	CPLEX11.0	LH_1	LH_2	CPLEX11.0
	15	5	5.66	4	30	1	30	11.67	86.67	96.67	62168.60	6511.90	10800
4 × 4	30	9	11.49	8	30	—	30	21.66	73.33	—	38385.88	6619.67	10800
	59	14	22.98	13	30	5	30	39.08	56.67	83.33	40002.87	7517.26	10800
	27	5	21.82	4	30	—	30	77.09	86.67	—	200198.16	18238.83	10800
5 × 5	55	9	12.40	5	30	—	30	27.43	83.33	—	188755.22	15909.70	10800
	104	15	19.50	14	30	—	30	23.10	53.33	—	347632.39	18681.06	10800
	39	5	23.41	5	30	—	—	78.64	83.33	—	487534.48	32267.35	—
6 × 6	77	10	15.15	9	30	—	—	33.99	70.00	—	693093.45	32391.73	—
	144	15	20.64	15	30	—	—	27.32	50.00	—	682000.64	37063.98	—
	52	5	30	5	30	—	—	83.33	83.33	—	1146504.17	72779.46	—
7 × 7	107	10	30	10	30	—	—	66.67	66.67	—	105178.14	58195.60	—
	198	5	30	10	30	—	—	83.33	66.67	—	88777.79	60382.95	—
	69	—	—	5	30	—	—	—	83.33	—	—	145335.12	—
8 × 8	142	—	—	9	30	—	—	—	70.00	—	—	283443.68	—
	261	—	—	18	30	—	—	—	40.00	—	—	186621.23	—
	85	—	—	5	30	—	—	—	83.33	—	—	86612.24	—
9 × 9	177	—	—	10	30	—	—	—	66.67	—	—	86493.82	—
	328	—	—	14	30	—	—	—	53.33	—	—	86868.76	—
Average:		8.92	20.25	9.06	30	3.00	30	47.78	69.81	90.00	340019.32	68996.35	10800

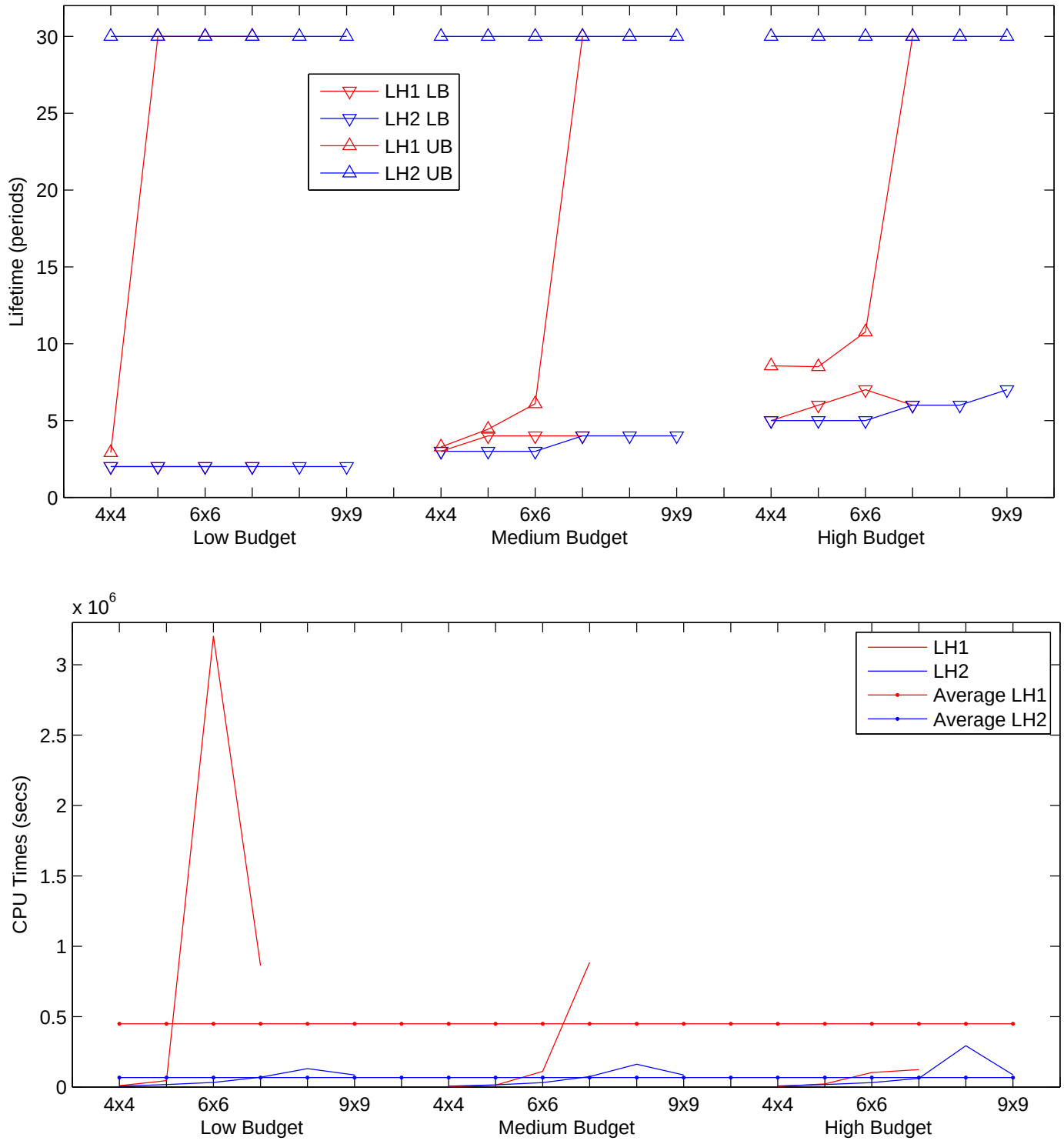

Figure 6.1. Comparison of LH_1 and LH_2 at low energy level

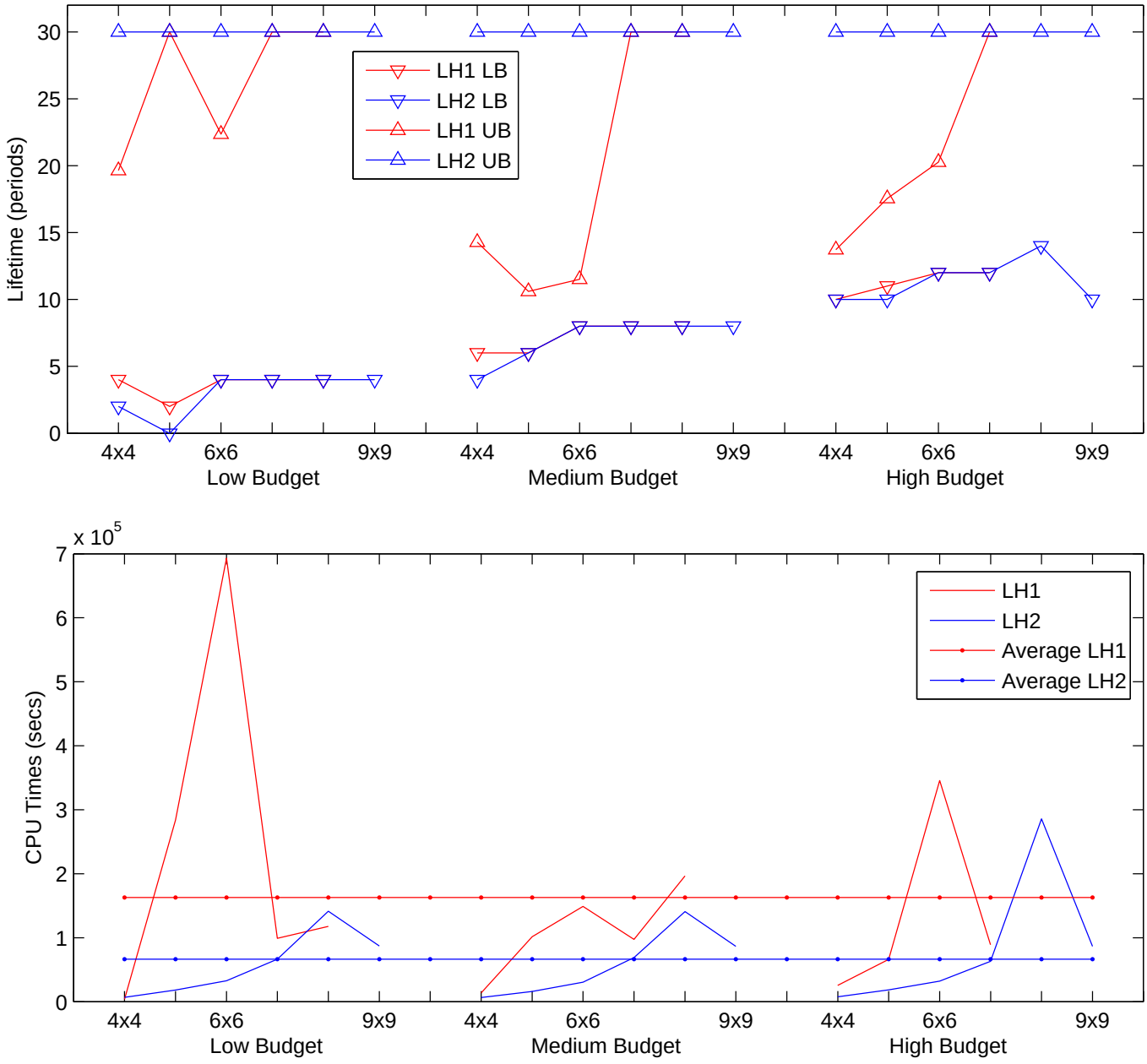

Figure 6.2. Comparison of LH_1 and LH_2 at medium energy level

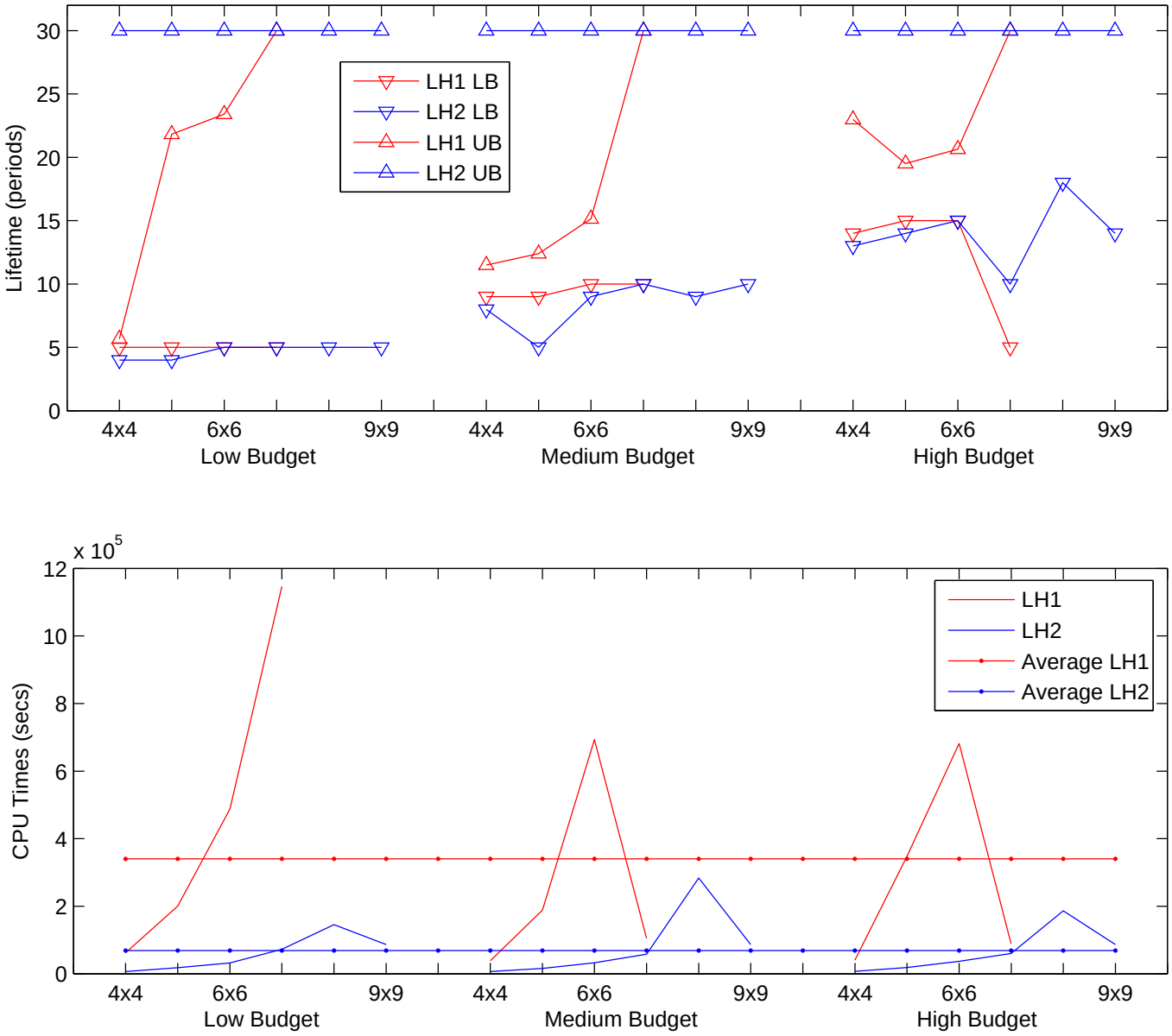

Figure 6.3. Comparison of LH_1 and LH_2 at high energy level

6.4. Results for Sink Location, Sensor Placement, Scheduling and Routing Problem with Connectivity Restrictions

In this section we assess the performance of the Local Search (LS) and Tabu Search (TS) heuristics developed in Chapter 5. The feasible solution generation algorithms GH and DH are used in both of them. The energy levels selected for the experiments are summarized in Table 6.7 and the formulas to calculate the budget levels are given in Table 6.8 which are similar to the properties of real sensors. The full energy capacity of a type-k sensor's battery represents its high energy level. The medium energy level is 2/3 of full battery energy and 1/3 of full battery energy refers to low energy level.

Table 6.7. Energy levels used in LS and TS runs

k	E_{low}	E_{medium}	E_{high}
1	19200	38400	57600
2	28800	57600	86400

Low budget level allows to deploy 75% of the points with type-1 and 25% with type-2 sensors, with medium budget level we can deploy 50% of the points with type-1 and 50% with type-2 sensors and with high level budget we can deploy 25% of the points with type-1 sensors and 75% with type-2 sensors.

Table 6.8. Formulas for the budget levels used in LS and TS runs

B_{low}	$\sum_{j \in \boldsymbol{N}} (0.75\ c_{j1} + 0.25\ c_{j2})$
B_{medium}	$\sum_{j \in \boldsymbol{N}} (0.50\ c_{j1} + 0.50\ c_{j2})$
B_{high}	$\sum_{j \in \boldsymbol{N}} (0.25\ c_{j1} + 0.75\ c_{j2}$

In all instances we aim to cover each point in the sensor field by at least two active sensors in period t. The network lifetime is maximized over a planning horizon $T = 400$.

First we compare the efficiency of the feasible solution generation algorithms GH

and DH to find a lower bound for the network lifetime. Their sensitivity to the number of sinks is also evaluated by conducting experiments with two and three sinks located randomly over the sensor field. The accuracy of the heuristics can be analyzed for the instances upto 5×5 grid network by using the results of brute force computations, since they show the optimal solution for the given instance, summarized in Table 6.9. As far as we can observe from the results that Discrimination Heuristic (DH) gives better lower bounds for the network lifetime than Greedy Heuristic (GH) but requires a little more time to find the solution on average.

The quality of the lower bounds obtained from algorithms GH and DH at one iteration and the corresponding computational times can be seen from Figure 6.4 - Figure 6.9 for each energy level. In order to see the performance of algorithms GH and DH with respect to the number of sinks, we take the difference of the lower bounds obtained for three sinks and two sinks. Moreover, the lower bounds found by algorithms GH and DH with the same number of sinks are evaluated through taking the difference of the lower bounds calculated by each algorithm for two sinks (and three sinks). As shown in the figures, for every energy level GH cannot improve the lifetime as the number of sinks increases. On the other hand, on average DH improves the lifetime as the number of sinks increases for low and high energy levels while it cannot develop the average lifetime for medium energy level due to its worse performance at medium and high budget levels. When there are two sinks in the sensor field, algorithm DH gives better lower bounds than GH for all energy levels except low energy level. However, DH outperforms at all energy levels when we have three sinks.

Regarding with the computational times, at all energy levels the average CPU times of GH both for two and three sinks are larger than the average CPU time of DH with any number of sinks. More specifically, algorithm GH requires more time on the average to find a lower bound with three sinks than it does for two sinks for low and high energy levels. At medium energy level, average CPU time for two sinks is larger than the one of three sinks. On the other side, algorithm DH uses more time on the average for three sinks than it does for two sinks for all energy levels.

The results indicate that algorithm DH is performing better than algorithm GH at one iteration in terms of lower bound on the network lifetime and the computational time. The weakness of algorithm GH may be due to its inefficient budget usage strategy. According to algorithm in Figure 4.3, the coverage and budget constraints are tried to be satisfied as more periods as possible within the planning horizon. For this purpose, in the case of requirement new sensors are deployed and activated which consumes the budget. After this step, the algorithm settles the connectivity of the network and make the sink assignments for the active sensors. Deployment of new sensors can again be necessary, especially for large size sensor fields, to provide the connectivity constraints. However, this needs available budget which is highly depleted at first step. Therefore, even though we have a lot of periods in which the coverage constraints are satisfied, we can provide connectivity for few periods only since there is not enough budget to deploy new sensors. On the contrary, the algorithm DH deals each period seperately to satisfy the coverage, budget and connectivity constraints which manages the budget more efficiently. As a result, algorithm GH reports a weak lower bound for the network lifetime than algorithm DH does. Besides, algorithm GH has to work on periods which will not be within the lifetime at the end of the algorithm which increases the computational time unnecessarily.

Table 6.9. Comparison of brute force results of Greedy and Discrimination Heuristics

			Optimum Lifetime				CPU Time (s)			
			S = 2		S = 3		S = 2		S = 3	
n × n	E_k	B	*GH*	*DH*	*GH*	*DH*	*GH*	*DH*	*GH*	*DH*
		15	76	88	76	88	47.48	52.62	261.64	275.32
4 × 4	low	30	101	106	101	106	34.24	71.57	200.69	339.39
		59	101	106	101	106	72.68	82.96	339.00	260.43
		15	154	176	154	176	123.00	127.96	354.44	353.32
4 × 4	medium	30	205	215	205	215	68.07	147.13	249.36	402.23
		59	205	215	205	215	147.56	167.21	457.15	444.83
		15	232	267	232	267	191.12	116.34	449.53	591.82
4 × 4	high	30	309	324	309	324	55.14	139.76	407.06	612.33
		59	309	324	309	324	134.25	145.06	675.81	684.86
		27	101	101	101	101	245.94	239.58	1979.12	1855.26
5 × 5	low	55	101	101	101	101	242.27	256.02	2041.45	2028.69
		104	114	114	114	114	288.79	283.91	2383.99	2298.60
		27	205	202	205	202	474.15	534.60	5716.51	5804.42
5 × 5	medium	55	205	202	205	202	522.20	509.89	5646.97	6235.28
		104	231	228	231	228	570.93	551.34	5041.87	6507.82
		27	309	304	309	304	744.64	764.41	5708.66	5655.69
5 × 5	high	55	309	304	309	304	768.89	723.32	8657.64	7490.95
		104	348	342	348	342	806.04	876.44	8161.54	7016.95
Average:			200.8	206.6	200.8	206.6	307.63	321.67	2707.36	2713.79

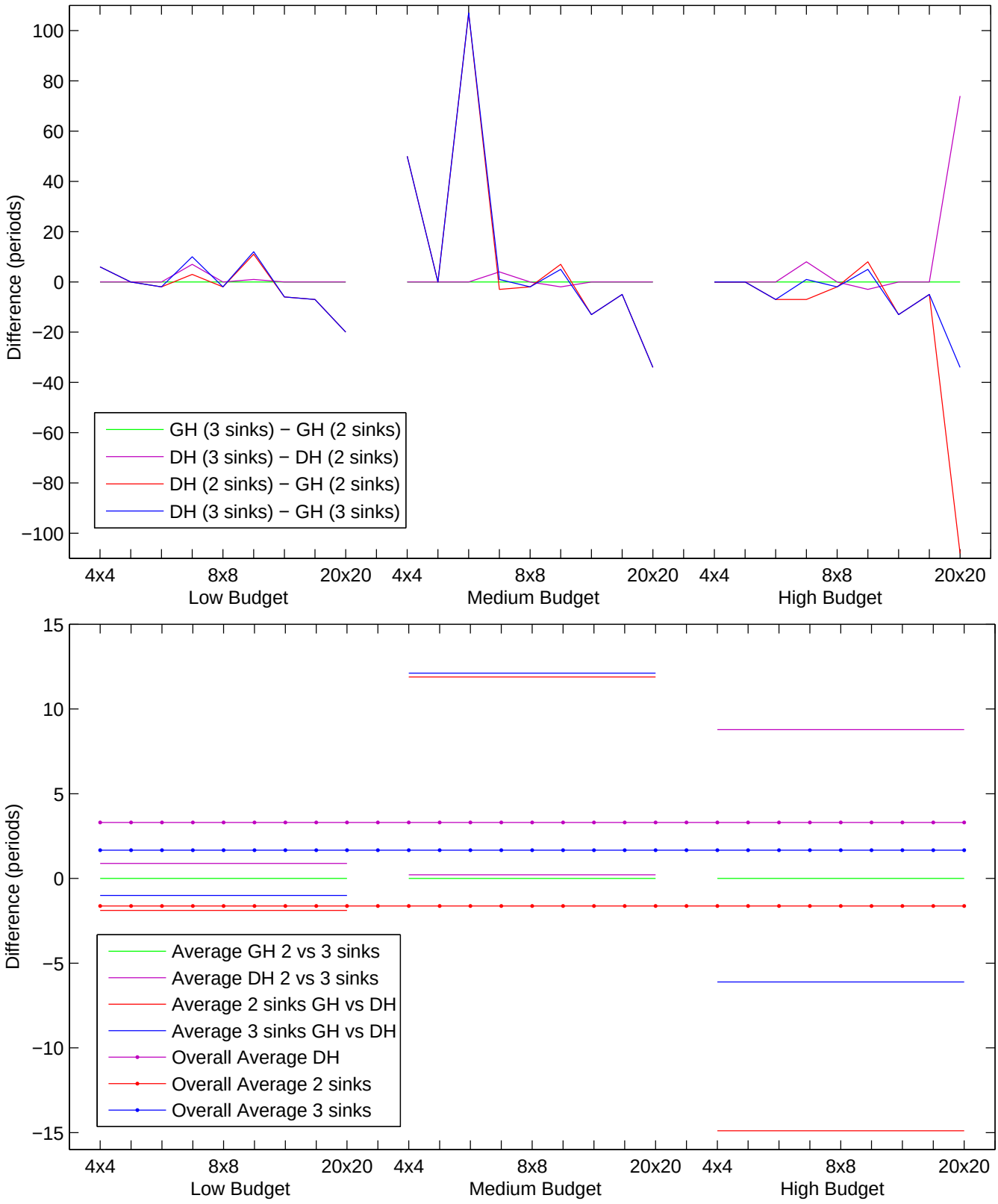

Figure 6.4. Comparison of algorithms GH and DH at low energy level

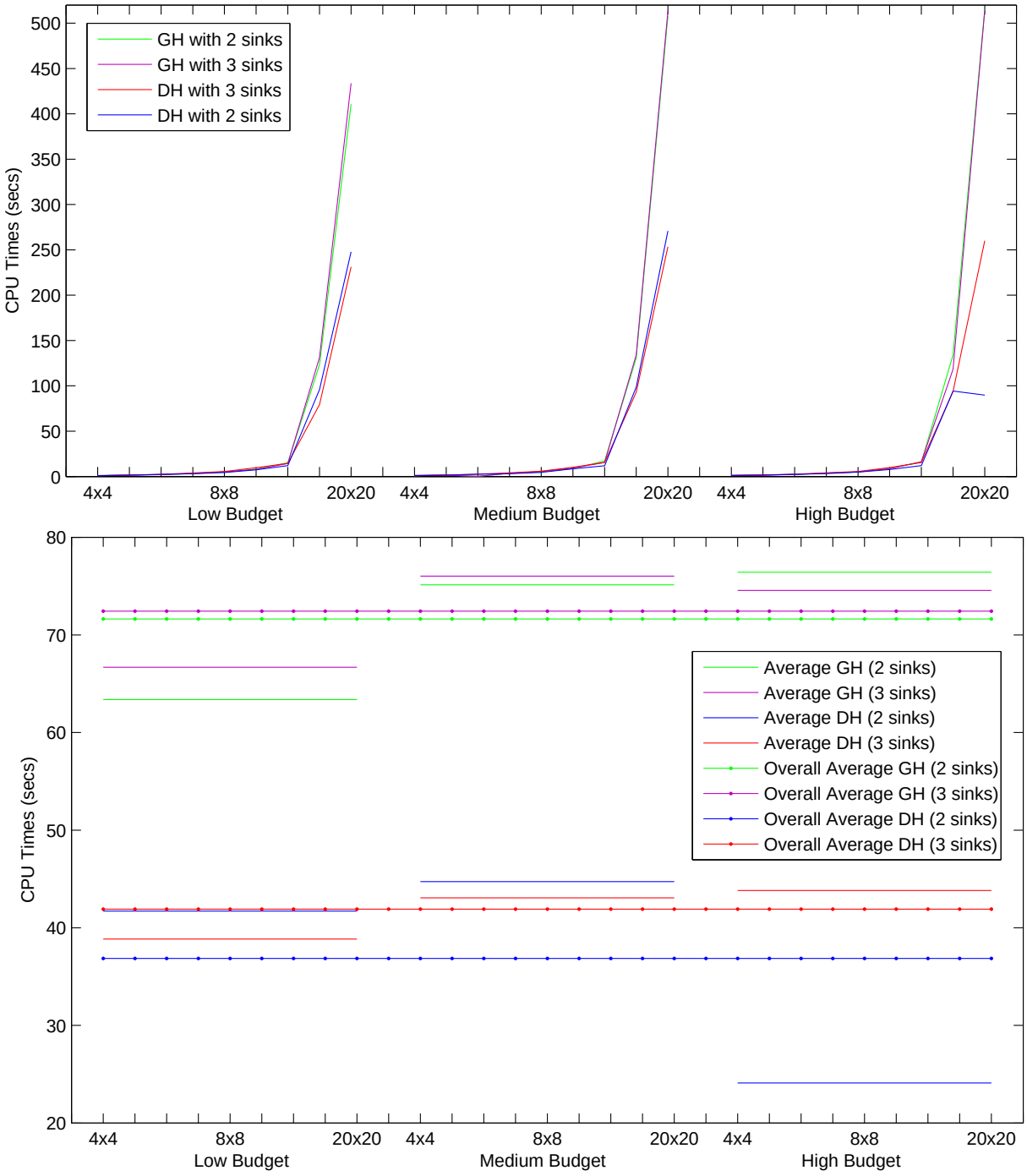

Figure 6.5. CPU times for algorithms GH and DH at low energy level

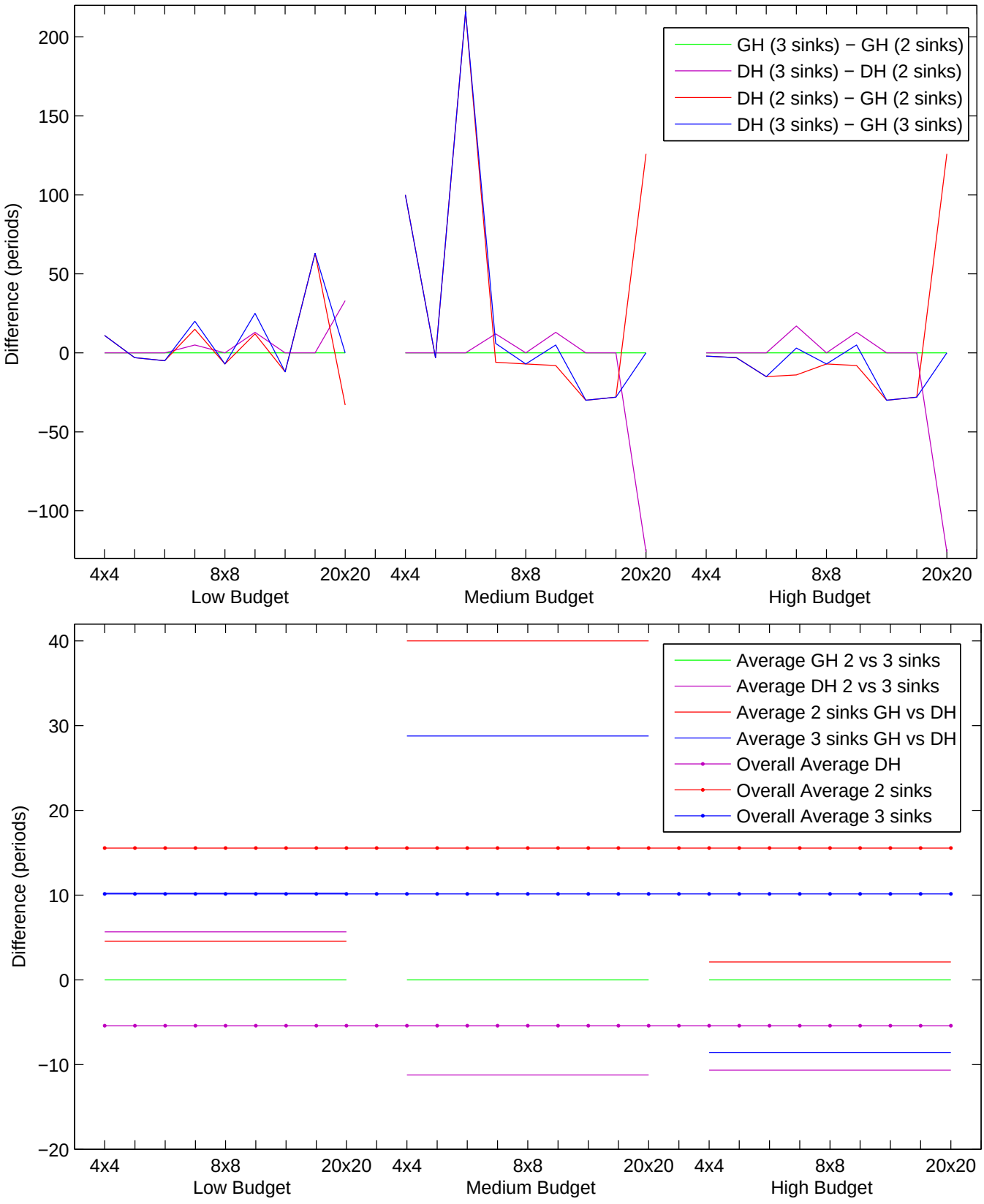

Figure 6.6. Comparison of algorithms GH and DH at medium energy level

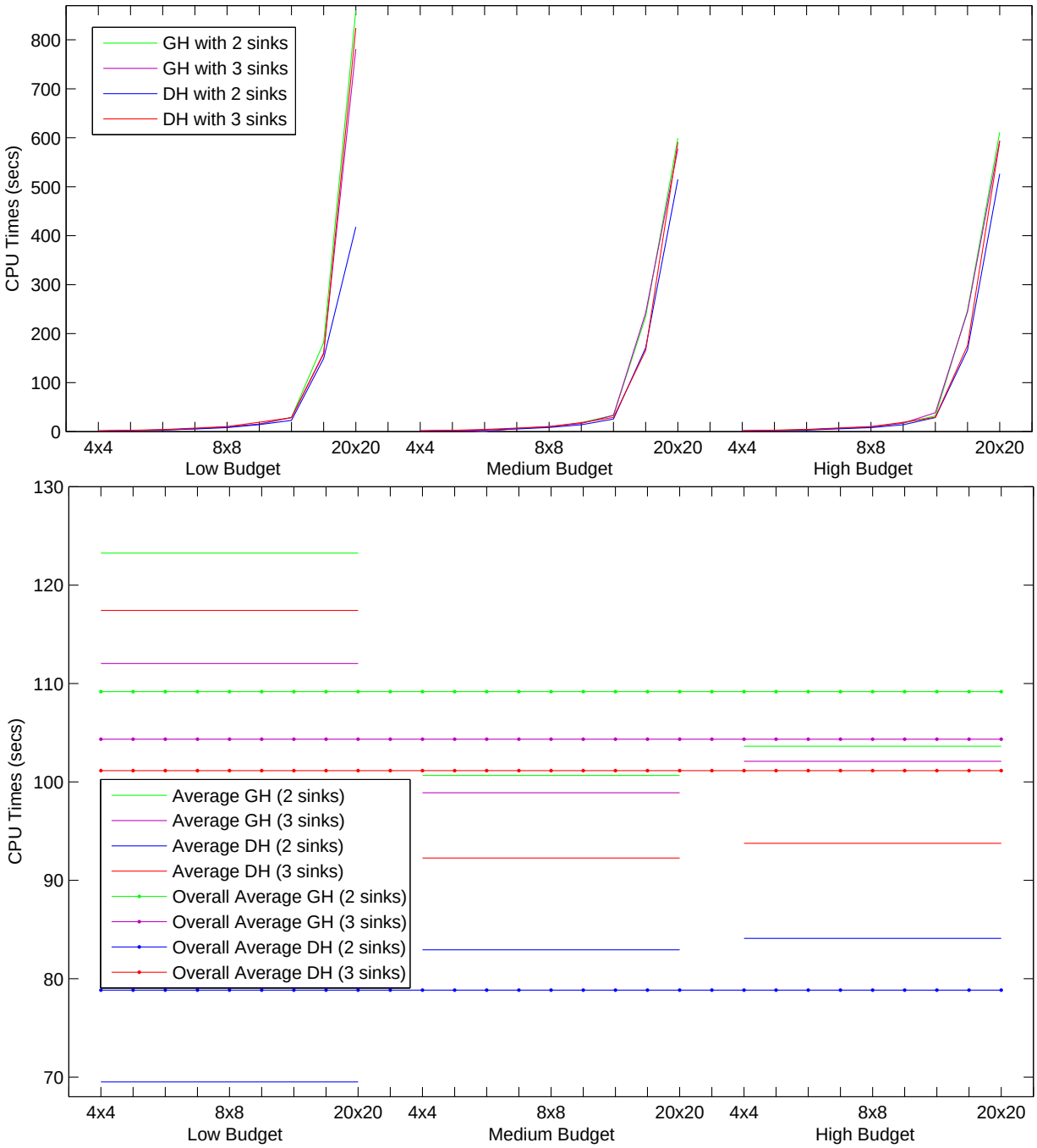

Figure 6.7. CPU times for algorithms GH and DH at medium energy level

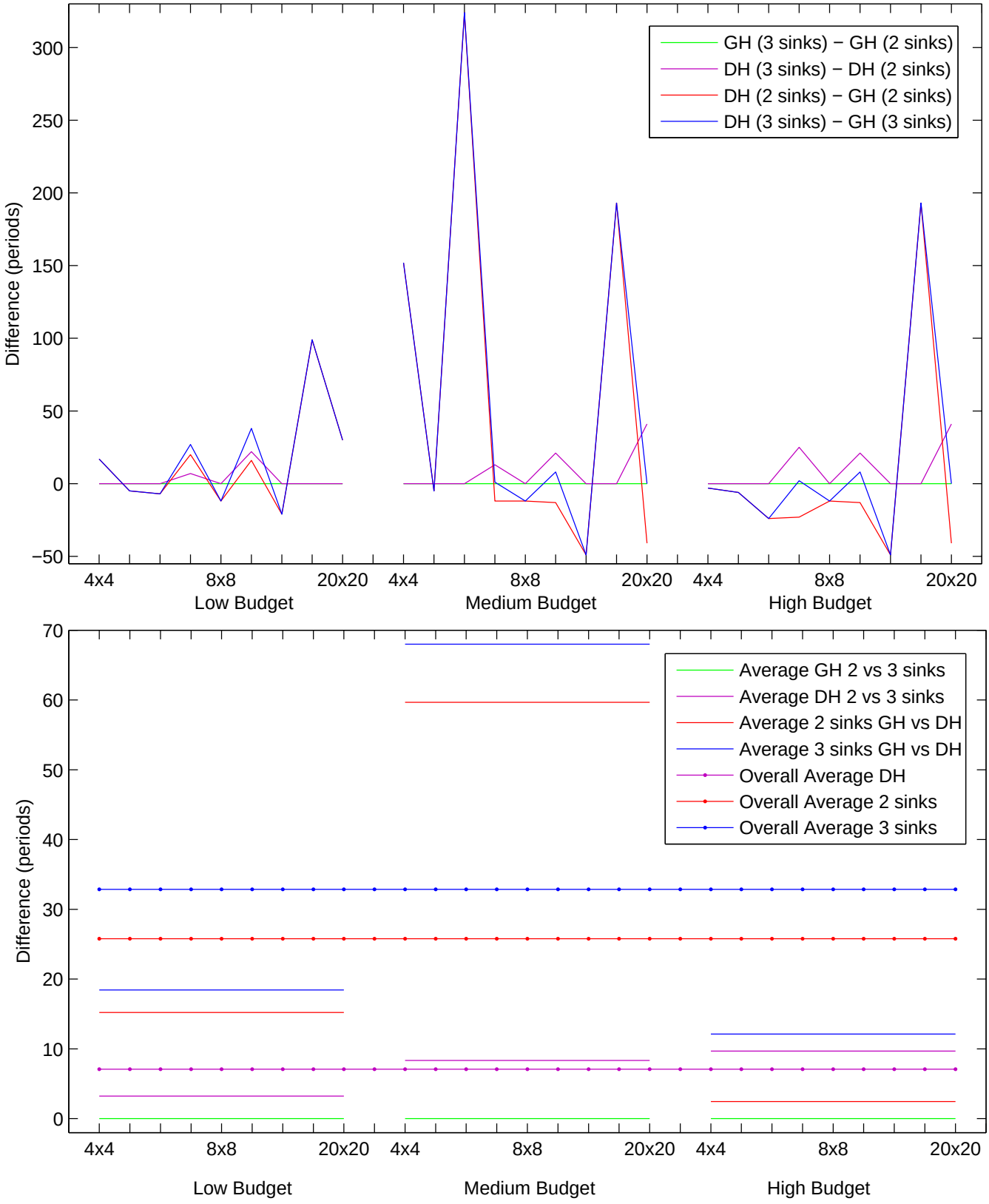

Figure 6.8. Comparison of algorithms GH and DH at high energy level

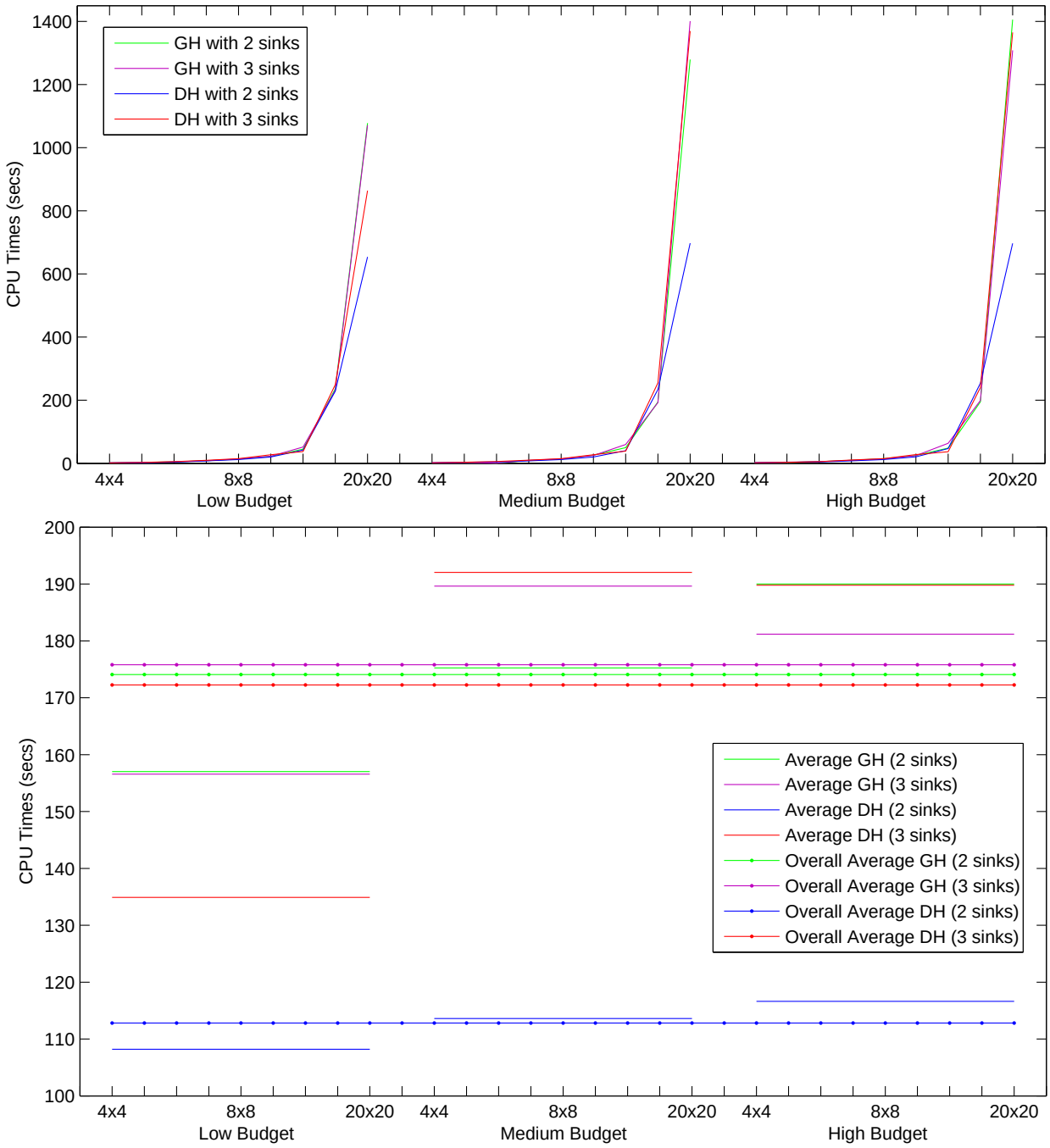

Figure 6.9. CPU times for algorithms GH and DH at high energy level

We conduct experiments for each of the search algorithms LS and TS with algorithms GH and DH seperately under different number of sinks. Our intention is to understand given the number of sinks in the network and the search algorithm, which feasible solution generation algorithm gives good lower bounds for the network lifetime. The performance of the search algorithms LS and TS are compared for a given number of sinks with their best executing heuristic. Both of the search algorithms run for at most $iterlim = 100$ iterations while the best lower bound is updated at least once in any consecutive $NI = 20$ iterations. The length of the tabu list, i.e. *tabutenure*, is assumed to be 10 for the algorithm TS. The percentage of the size of a s-swap neighborhood that is explored by each search algorithm can be summarized in Table 6.10.

Table 6.10. The explored percentage of a s-swap neighborhood by LS and TS

	LS			TS		
s	1	2	3	1	2	3
P_s (%)	20	40	40	100	20	10

The search algorithms we propose are generating lower bounds for the network lifetime. Then the accuracy of the search algorithms can be done through comparison of the best known lower bound with the one found by the search algorithm. The optimum lower bound values with feasible solution generation algorithms GH and DH are given by Brute Force (BF) algorithm for instances upto 5×5 as in Table 6.9. Besides, the algorithms LS and TS find lower bound values. Therefore, given a feasible solution generation algorithm, say $FS \in \{GH,\ DH\}$, and a search algorithm, say $SA \in \{LS,\ TS\}$, the accuracy of the lower bound generated with the algorithm SA using algorithm FS can be given as

$$100 \times \frac{\max_{FS}\{BF_{FS}, LS_{FS}, TS_{FS}\} - SA_{FS}}{\max_{FS}\{BF_{FS}, LS_{FS}, TS_{FS}\}} \ . \tag{6.2}$$

The results summarized in Table 6.11 and Table 6.12 list the lower bounds for network lifetime of the algorithms LS and TS with subalgorithm GH at low energy

level. We can observe that on the average, applying a search algorithm over the possible locations of sinks improves the lifetime found by algorithm GH that runs for only one iteration. However, at low energy level we cannot upgrade the network lifetime when we search the solution space with algorithm TS instead of algorithm LS. Besides, increasing the number of sinks in the sensor field does not help the network to operate longer periods. We can see the results for medium energy level in Table 6.13 and Table 6.14, and for high energy level in Table 6.15 and Table 6.16. As the initial energy level of the sensors gets higher, the average lower bound for the network lifetime develops as expected. On the other hand, even though we can find better lower bounds when we search for alternative locations for the sinks, we cannot expand the network lifetime by introducing additional sinks to the sensor field for both of the search algorithms. These results are similar to low energy level case, from which we conclude that algorithm GH is not sensitive to the increment in the number of sinks from two to three in the network.

The accuracies of the search algorithms are also listed in the tables from which we can see with the same number of sinks, both of the algorithms LS and TS have identical accuracy levels for all energy levels. Comparing the average accuracies of the algorithms LS and TS under different number of sinks, we can conclude that at low and medium energy levels the search algorithms appear to be more accurate with two sinks than with three sinks. However, at high energy level the search algorithms become more efficient with three sinks. This means, with the increasing number of sinks in the network, both of the search algorithms generate better lower bounds that are closer to the best known solution when the sensors have the highest initial battery energy. However, this is a misleading interpretation of the results since the average for three sinks is calculated with less number of instances. The average accuracies for the common 21 instances including 10×10 network, which are given in "Adjusted Avg" row of the tables, reveal that at high energy level there is no difference among the accuracies of two and three-sink cases.

Table 6.11. Results for LS and TS with GH at low energy level

		Lower Bound						Accuracy			
		S = 2			S = 3			S = 2		S = 3	
n × n	B	GH	LS_{GH}	TS_{GH}	GH	LS_{GH}	TS_{GH}	LS_{GH}	TS_{GH}	LS_{GH}	TS_{GH}
	111	76	76	76	76	76	76	13.64	13.64	13.64	13.64
4 × 4	126	51	101	101	51	101	101	4.72	4.72	4.72	4.72
	136	101	101	101	101	101	101	4.72	4.72	4.72	4.72
	197	101	101	101	101	101	101	0	0	0	0
5 × 5	223	101	101	101	101	101	101	0	0	0	0
	238	114	114	114	114	114	114	0	0	0	0
	275	101	101	101	101	101	101	2.88	2.88	2.88	2.88
6 × 6	312	0	114	114	0	114	114	0	0	0	0
	331	114	114	114	114	114	114	0	0	0	0
	375	101	101	101	101	101	101	9.01	9.01	9.01	9.01
7 × 7	426	114	114	114	114	114	114	2.56	2.56	2.56	2.56
	453	118	118	118	118	118	118	4.84	4.84	4.84	4.84
	497	101	101	101	101	101	101	0	0	0	0
8 × 8	562	101	101	101	101	101	101	0	0	0	0
	593	101	101	101	101	101	101	0	0	0	0

Table 6.12. Results for LS and TS with GH at low energy level (cont)

		Lower Bound						Accuracy			
		S = 2			S = 3			S = 2		S = 3	
n × n	B	GH	LS_{GH}	TS_{GH}	GH	LS_{GH}	TS_{GH}	LS_{GH}	TS_{GH}	LS_{GH}	TS_{GH}
	622	96	106	106	96	106	106	1.85	1.85	1.85	1.85
9 × 9	705	106	106	106	106	106	106	6.19	6.19	6.19	6.19
	745	106	106	106	106	106	106	7.83	7.83	7.83	7.83
	754	114	114	114	114	114	114	0	0	0	0
10 × 10	855	126	126	126	126	126	126	0	0	0	0
	903	126	126	126	126	126	126	0	0	0	0
	1686	114	114	114	114	—	—	0	0	—	—
15 × 15	1918	119	126	126	119	—	—	0	0	—	—
	2024	119	126	126	119	—	—	0	0	—	—
	2958	116	—	—	116	—	—	—	—	—	—
20 × 20	3369	139	—	—	139	—	—	—	—	—	—
	3564	139	—	—	139	—	—	—	—	—	—
Average:		104.2	108.7	108.7	104.2	106.8	106.8	2.43	2.43	2.77	2.77
Adjusted Avg:		98.5	106.8	106.8	98.5	106.8	106.8	2.77	2.77	2.77	2.77

Table 6.13. Results for LS and TS with GH at medium energy level

		Lower Bound						Accuracy			
		S = 2			S = 3			S = 2		S = 3	
n × n	B	GH	LS_{GH}	TS_{GH}	GH	LS_{GH}	TS_{GH}	LS_{GH}	TS_{GH}	LS_{GH}	TS_{GH}
	111	154	154	154	154	154	154	12.50	12.50	12.50	12.50
4 × 4	126	103	205	205	103	205	205	4.65	4.65	4.65	4.65
	136	205	205	205	205	205	205	4.65	4.65	4.65	4.65
	197	205	205	205	205	205	205	0	0	0	0
5 × 5	223	205	205	205	205	205	205	0	0	0	0
	238	231	231	231	231	231	231	0	0	0	0
	275	205	205	205	205	205	205	2.38	2.38	2.38	2.38
6 × 6	312	0	231	231	0	231	231	0	0	0	0
	331	231	231	231	231	231	231	0	0	28.92	28.92
	375	205	205	205	205	205	205	8.89	8.89	36.92	36.92
7 × 7	426	231	231	231	231	231	231	2.53	2.53	2.53	2.53
	453	239	239	239	239	239	239	4.40	4.40	4.40	4.40
	497	205	205	205	205	205	205	0	0	0	0
8 × 8	562	205	205	205	205	205	205	0	0	0	0
	593	205	205	205	205	205	205	0	0	0	0

Table 6.14. Results for LS and TS with GH at medium energy level (cont)

		Lower Bound						Accuracy			
		S = 2			S = 3			S = 2		S = 3	
n × n	B	GH	LS_{GH}	TS_{GH}	GH	LS_{GH}	TS_{GH}	LS_{GH}	TS_{GH}	LS_{GH}	TS_{GH}
	622	194	214	214	194	214	214	2.28	2.28	2.28	2.28
9 × 9	705	214	214	214	214	214	214	4.89	4.89	4.89	4.89
	745	214	214	214	214	214	214	6.14	6.14	6.14	6.14
	754	231	231	231	231	231	231	0	0	0	0
10 × 10	855	256	256	256	256	256	256	0	0	0	0
	903	256	256	256	256	256	256	4.48	4.48	4.48	4.48
	1686	147	147	147	147	—	—	31.31	31.31	—	—
15 × 15	1918	256	256	256	256	—	—	0	0	—	—
	2024	256	256	256	256	—	—	0	0	—	—
	2958	231	—	—	231	—	—	—	—	—	—
20 × 20	3369	103	—	—	103	—	—	—	—	—	—
	3564	103	—	—	103	—	—	—	—	—	—
Average:		195.9	216.9	216.9	195.9	216.5	216.5	3.71	3.71	2.75	2.75
Adjusted Avg:		199.7	216.5	216.5	199.7	216.5	216.5	2.75	2.75	2.75	2.75

Table 6.15. Results for LS and TS with GH at high energy level

		Lower Bound						Accuracy			
		S = 2			S = 3			S = 2		S = 3	
n × n	B	GH	LS_{GH}	TS_{GH}	GH	LS_{GH}	TS_{GH}	LS_{GH}	TS_{GH}	LS_{GH}	TS_{GH}
	111	232	232	232	154	232	232	13.11	13.11	13.11	13.11
4 × 4	126	154	309	309	154	309	309	4.63	4.63	4.63	4.63
	136	309	309	309	309	309	309	4.63	4.63	4.63	4.63
	197	309	309	309	309	309	309	0	0	0	0
5 × 5	223	309	309	309	309	309	309	0	0	0	0
	238	348	348	348	348	348	348	0	0	0	0
	275	309	309	309	309	309	309	2.83	2.83	2.83	2.83
6 × 6	312	0	348	348	0	348	348	0	0	0	0
	331	348	348	348	348	348	348	0	0	0	0
	375	309	309	309	309	309	309	8.04	8.04	8.04	8.04
7 × 7	426	348	348	348	348	348	348	0.57	0.57	0.57	0.57
	453	359	359	359	359	359	359	4.01	4.01	4.01	4.01
	497	309	309	309	309	309	309	0	0	0	0
8 × 8	562	309	309	309	309	309	309	0	0	0	0
	593	309	309	309	309	309	309	0	0	0	0

Table 6.16. Results for LS and TS with GH at high energy level (cont)

		Lower Bound						Accuracy			
		S = 2			S = 3			S = 2		S = 3	
n × n	B	GH	LS_{GH}	TS_{GH}	GH	LS_{GH}	TS_{GH}	LS_{GH}	TS_{GH}	LS_{GH}	TS_{GH}
	622	292	323	323	292	323	323	2.12	2.12	2.12	2.12
9 × 9	705	323	323	323	323	323	323	4.44	4.44	4.44	4.44
	745	323	323	323	323	323	323	6.65	6.65	6.65	6.65
	754	348	348	348	348	348	348	0	0	0	0
10 × 10	855	386	386	386	386	386	386	0	0	0	0
	903	386	386	386	386	386	386	3.50	3.50	3.50	3.50
	1686	216	216	216	216	—	—	32.71	32.71	—	—
15 × 15	1918	149	384	384	149	—	—	0	0	—	—
	2024	149	384	384	149	—	—	0	0	—	—
	2958	262	—	—	262	—	—	—	—	—	—
20 × 20	3369	384	—	—	384	—	—	—	—	—	—
	3564	384	—	—	384	—	—	—	—	—	—
Average:		291.2	326.5	326.5	326.5	326.3	326.3	3.63	3.63	2.60	2.60
Adjusted Avg:		300.9	326.3	326.3	300.9	326.3	326.3	2.60	2.60	2.60	2.60

Computational times of the search algorithms LS and TS with subalgorithm GH at different energy levels are given in the Table 6.17 - Table 6.19. At all energy levels, the average computational time for algorithm LS is less than the average required time for algorithm TS with two sinks. When there are three sinks in the sensor field, both of the search algorithms consume almost same amount of time on average to find a lower bound. We expect that a search algortihm terminates in longer time as there are more sinks in the network since the size of a s-swap neighborhood increases which requires more iterations to search a certain percantage of the neighborhood. The reported average computation times in the tables do not agree with the expectation. This is again due to the number of instances that are used to calculate the average running times of the algorithms. Our expectation appears to be valid if we calculate the average computation times for the same number of instances as given in "Adjusted Avg" row.

Table 6.17. CPU times for LS and TS with GH at low energy level

		CPU Times (s)					
		S = 2			S = 3		
n × n	B	GH	LS_{GH}	TS_{GH}	GH	LS_{GH}	TS_{GH}
	111	1.03	12.02	20.00	0.94	25.65	21.33
4 × 4	126	0.91	5.90	24.06	0.85	20.81	17.59
	136	1.08	14.55	23.19	1.02	33.06	24.96
	197	1.55	28.96	60.73	1.35	61.72	52.48
5 × 5	223	1.54	35.86	43.39	1.36	62.07	57.89
	238	1.67	37.92	48.14	1.46	72.19	66.41
	275	2.12	63.71	80.16	2.14	116.82	101.35
6 × 6	312	0.24	38.21	55.25	0.24	70.61	72.79
	331	2.29	76.53	96.00	2.30	131.07	123.74
	375	3.26	110.35	203.74	3.45	205.90	175.34
7 × 7	426	3.62	131.93	222.26	3.63	227.27	231.41
	453	3.71	138.76	223.42	3.71	219.51	246.73
	497	5.13	197.90	399.81	5.10	331.42	308.68
8 × 8	562	5.11	182.18	372.94	5.57	339.64	342.13
	593	5.10	179.27	432.10	5.08	347.15	327.88
	622	8.14	318.15	569.53	7.94	566.04	576.30
9 × 9	705	8.76	330.29	592.56	9.34	565.43	483.34
	745	8.99	356.49	647.16	8.59	508.81	573.11
	754	15.17	592.20	638.12	14.31	961.69	912.98
10 × 10	855	17.22	645.86	730.32	15.50	1086.73	1030.04
	903	15.61	684.19	754.95	16.32	1047.74	1051.38
	1686	123.39	3908.05	4132.92	131.33	—	—
15 × 15	1918	130.54	3662.65	5050.96	133.99	—	—
	2024	133.99	4087.74	3929.83	118.23	—	—
	2958	410.57	—	—	433.66	—	—
20 × 20	3369	508.93	—	—	513.67	—	—
	3564	515.32	—	—	514.33	—	—
Average:		71.64	659.57	806.19	72.42	333.40	323.71
Adjusted Avg:		5.35	198.63	297.04	5.25	333.40	323.71

Table 6.18. CPU times for LS and TS with GH at medium energy level

		CPU Times (s)					
		S = 2			S = 3		
n × n	B	GH	LS_{GH}	TS_{GH}	GH	LS_{GH}	TS_{GH}
	111	1.38	33.40	42.00	1.27	54.40	37.80
4 × 4	126	1.17	15.15	35.57	1.02	40.50	33.66
	136	1.64	31.73	48.16	1.41	67.60	48.91
	197	2.31	71.91	88.44	2.10	129.53	115.88
5 × 5	223	2.49	72.46	84.56	2.11	130.82	125.26
	238	2.47	86.47	89.41	2.30	142.24	127.68
	275	3.57	140.12	170.34	3.36	243.74	209.41
6 × 6	312	0.38	79.62	112.81	0.39	144.51	140.72
	331	3.94	149.54	203.10	3.82	270.16	261.82
	375	5.79	221.46	402.70	6.12	390.18	385.45
7 × 7	426	6.58	262.85	391.76	6.40	454.80	449.61
	453	6.65	261.33	475.15	6.68	447.26	416.68
	497	9.35	336.71	837.55	9.52	833.64	660.67
8 × 8	562	9.45	387.42	864.67	10.19	687.73	667.64
	593	9.36	385.68	807.40	9.69	683.16	690.73
	622	15.13	713.60	1263.79	15.54	1148.69	877.02
9 × 9	705	17.97	757.04	1445.47	16.62	1181.77	1092.68
	745	17.43	806.68	1417.14	17.21	1137.72	1164.73
	754	29.10	1187.48	1406.82	28.58	1839.75	1972.17
10 × 10	855	33.61	1253.36	1501.37	33.04	1834.40	2371.13
	903	32.10	1457.29	1690.32	38.42	1977.02	2016.71
	1686	181.94	5101.14	5029.65	160.45	—	—
15 × 15	1918	234.89	7431.15	9277.23	242.56	—	—
	2024	247.48	8116.57	7844.08	245.04	—	—
	2958	860.69	—	—	781.32	—	—
20 × 20	3369	599.30	—	—	577.80	—	—
	3564	611.40	—	—	594.23	—	—
Average:		109.17	1223.34	1480.31	104.34	659.03	660.30
Adjusted Avg:		10.09	414.82	636.98	10.28	659.03	660.30

Table 6.19. CPU times for LS and TS with GH at high energy level

		CPU Times (s)					
		S = 2			S = 3		
n × n	B	GH	LS_{GH}	TS_{GH}	GH	LS_{GH}	TS_{GH}
	111	1.78	48.12	71.76	1.53	66.46	53.37
4 × 4	126	1.38	27.59	43.56	1.26	55.15	62.74
	136	2.09	54.78	77.61	1.98	98.46	97.33
	197	2.93	105.02	121.17	2.78	179.30	177.26
5 × 5	223	2.88	110.05	128.27	2.71	176.55	182.06
	238	3.20	112.37	151.02	2.93	208.90	186.48
	275	5.04	209.92	353.86	5.06	357.50	364.50
6 × 6	312	0.52	116.23	204.44	0.54	209.78	221.98
	331	5.55	216.48	372.17	5.32	375.07	380.39
	375	8.28	344.10	552.68	8.59	586.56	612.73
7 × 7	426	9.30	390.28	675.48	9.74	680.38	553.72
	453	9.65	397.23	802.93	9.88	687.01	688.69
	497	13.82	560.20	1024.72	14.23	1020.50	987.16
8 × 8	562	13.66	606.67	992.74	13.91	1104.49	936.43
	593	14.97	643.75	988.33	14.24	958.21	958.54
	622	22.16	911.94	1982.59	22.84	1706.48	1448.06
9 × 9	705	25.87	1087.93	1724.96	24.90	1761.44	1740.35
	745	25.19	1101.84	1583.66	24.80	1714.16	1750.33
	754	45.12	1598.45	1863.37	51.72	2778.27	2833.14
10 × 10	855	49.80	2176.86	2380.09	59.82	3065.59	2870.14
	903	48.64	2055.01	2543.26	63.59	2836.70	2879.49
	1686	236.10	3100.83	6566.13	231.29	—	—
15 × 15	1918	194.41	5545.24	6913.92	192.75	—	—
	2024	194.99	6045.50	7104.65	199.28	—	—
	2958	1078.04	—	—	1071.01	—	—
20 × 20	3369	1279.35	—	—	1401.16	—	—
	3564	1405.70	—	—	1308.64	—	—
Average:		174.09	1148.60	1634.31	175.80	982.71	951.66
Adjusted Avg:		14.85	613.09	887.56	16.30	982.71	951.66

The lifetimes obtained from algorithms LS and TS with subalgorithm DH at low energy level are given in Table 6.20 and Table 6.21. Contrary to algorithm GH, algorithm DH is sensitive to the number of sinks in the sensor field which can be observed from the average lifetimes for two and three sinks reported for the algorithm DH that runs for one iteration. We observe from the average lifetime values that exploring the candidate locations for the sinks with a search algorithm helps to find good solutions. However, the average performances of the search algorithms LS and TS do not expand with the increasing number of sinks. This is due to the weak performance of the search algorithms at some budget levels of 4×4 instances. The lifetimes for 4×4 instances found by the algorithm LS drop off at all budget levels when the number of sinks in the network increases. In contrast with the intuiton, adding new sinks to the network does not necessarily evolve the network lifetime since we are making the use of a greedy selection criterion to make the sink assignments for active sensors. The effect of energy consumption in data transmission is not reflected to this selection rule. Therefore, an active sensor may be directed to a far sink even though there is another sink which is closer to itself. As a result, there can be unnecessary energy consumption, which declines the network lifetime, with the increased number of hops until the information reaches to the corresponding sink. On the other hand, algorithm TS seems to overcome the decreasing lifetime problem in 4×4 instances except for low budget case. This can be explained with the better diversification strategy of algorithm TS, which avoids stacking in a local optima for medium and high budget levels, when it is compared with algorithm LS. Since low budget level is not sufficient to accept the corresponding improving solution, it is not possible for algorithm TS to explore the different parts of the solution space. Finally, we can observe that algorithm TS outperforms algorithm LS for both number of sinks. At medium energy level, see Table 6.22 and Table 6.23, with two sinks both of the search algorithms give the same average lifetimes, whereas with three sinks algorithm TS surpasses algorithm LS. Similar to the low and medium energy levels, the performance of algorithm TS goes beyond the one of algorithm LS for both number of sinks at high energy level as can be seen in Table 6.24 and Table 6.25.

Table 6.20. Results for LS and TS with DH at low energy level

		Lower Bound						Accuracy			
		S = 2			S = 3			S = 2		S = 3	
n × n	B	DH	LS_{DH}	TS_{DH}	DH	LS_{DH}	TS_{DH}	LS_{DH}	TS_{DH}	LS_{DH}	TS_{DH}
	111	82	88	88	82	82	82	0	0	6.82	6.82
4 × 4	126	101	106	106	101	101	106	0	0	4.72	0
	136	101	106	106	101	101	106	0	0	4.72	0
	197	101	101	101	101	101	101	0	0	0	0
5 × 5	223	101	101	101	101	101	101	0	0	0	0
	238	114	114	114	114	114	114	0	0	0	0
	275	99	104	104	99	104	104	0	0	0	0
6 × 6	312	107	111	111	107	111	111	2.63	2.63	2.63	2.63
	331	107	111	111	107	111	111	2.63	2.63	2.63	2.63
	375	104	111	111	111	111	111	0	0	0	0
7 × 7	426	111	115	117	115	117	117	1.71	0	0	0
	453	111	124	124	119	124	124	0	0	0	0
	497	99	99	99	99	99	99	1.98	1.98	1.98	1.98
8 × 8	562	99	99	99	99	99	99	1.98	1.98	1.98	1.98
	593	99	99	99	99	99	99	1.98	1.98	1.98	1.98

Table 6.21. Results for LS and TS with DH at low energy level (cont)

		Lower Bound						Accuracy			
		S = 2			S = 3			S = 2		S = 3	
n × n	B	DH	LS_{DH}	TS_{DH}	DH	LS_{DH}	TS_{DH}	LS_{DH}	TS_{DH}	LS_{DH}	TS_{DH}
	622	107	108	108	108	108	108	0	0	0	0
9 × 9	705	113	113	113	111	113	113	0	0	0	0
	745	114	115	115	111	115	115	0	0	0	0
	754	108	108	108	108	108	108	5.26	5.26	5.26	5.26
10 × 10	855	113	120	120	113	120	120	4.76	4.76	4.76	4.76
	903	113	121	121	113	123	123	3.97	3.97	3.97	3.97
	1686	107	108	108	107	—	—	5.26	5.26	—	—
15 × 15	1918	114	114	114	114	—	—	9.52	9.52	—	—
	2024	114	114	114	114	—	—	9.52	9.52	—	—
	2958	96	—	—	96	—	—	—	—	—	—
20 × 20	3369	105	—	—	105	—	—	—	—	—	—
	3564	31	—	—	105	—	—	—	—	—	—
Average:		102.6	108.8	108.8	105.9	107.7	108.2	2.13	2.06	1.90	1.45
Adjusted Avg:		105.0	108.3	108.4	105.7	107.7	108.2	1.28	1.20	1.90	1.45

Table 6.22. Results for LS and TS with DH at medium energy level

		Lower Bound						Accuracy			
		S = 2			S = 3			S = 2		S = 3	
n × n	B	DH	LS_{DH}	TS_{DH}	DH	LS_{DH}	TS_{DH}	LS_{DH}	TS_{DH}	LS_{DH}	TS_{DH}
	111	165	176	176	165	165	165	0	0	6.25	6.25
4 × 4	126	203	215	215	203	203	215	0	0	5.58	0
	136	203	215	215	203	203	215	0	0	5.58	0
	197	202	202	202	202	202	202	1.46	1.46	1.46	1.46
5 × 5	223	202	202	202	202	202	202	1.46	1.46	1.46	1.46
	238	228	228	228	228	228	228	1.30	1.30	1.30	1.30
	275	200	210	210	200	210	210	0	0	0	0
6 × 6	312	216	225	225	216	225	225	2.60	2.60	2.60	2.60
	331	216	225	225	216	225	225	2.60	2.60	2.60	2.60
	375	220	225	225	225	225	225	0	0	0	0
7 × 7	426	225	237	237	237	237	237	0	0	0	0
	453	225	250	250	242	250	250	0	0	0	0
	497	198	198	198	198	198	198	3.41	3.41	3.41	3.41
8 × 8	562	198	198	198	198	198	198	3.41	3.41	3.41	3.41
	593	198	198	198	198	198	198	3.41	3.41	3.41	3.41

Table 6.23. Results for LS and TS with DH at medium energy level (cont)

		Lower Bound						Accuracy			
		S = 2			S = 3			S = 2		S = 3	
n × n	B	DH	LS_{DH}	TS_{DH}	DH	LS_{DH}	TS_{DH}	LS_{DH}	TS_{DH}	LS_{DH}	TS_{DH}
	622	206	219	219	219	219	219	0	0	0	0
9 × 9	705	206	225	225	219	225	225	0	0	0	0
	745	206	228	228	219	228	228	0	0	0	0
	754	219	219	219	219	219	219	5.19	5.19	5.19	5.19
10 × 10	855	226	244	244	226	245	245	4.69	4.69	4.30	4.30
	903	226	268	268	226	268	268	0	0	0	0
	1686	210	214	214	210	—	—	0	0	—	—
15 × 15	1918	228	228	228	228	—	—	10.94	10.94	—	—
	2024	228	228	228	228	—	—	10.94	10.94	—	—
	2958	198	—	—	231	—	—	—	—	—	—
20 × 20	3369	229	—	—	103	—	—	—	—	—	—
	3564	229	—	—	103	—	—	—	—	—	—
Average:		211.5	219.9	219.9	206.1	217.8	218.9	2.14	2.14	2.22	1.68
Adjusted Avg:		209.0	219.4	219.4	212.4	217.8	218.9	1.48	1.48	2.22	1.68

Table 6.24. Results for LS and TS with DH at high energy level

		Lower Bound						Accuracy			
		S = 2			S = 3			S = 2		S = 3	
n × n	B	DH	LS_{DH}	TS_{DH}	DH	LS_{DH}	TS_{DH}	LS_{DH}	TS_{DH}	LS_{DH}	TS_{DH}
	111	249	267	267	249	249	249	0	0	6.74	6.74
4 × 4	126	306	324	324	306	306	324	0	0	5.56	0
	136	306	324	324	306	306	324	0	0	5.56	0
	197	304	304	304	304	304	304	1.62	1.62	1.62	1.62
5 × 5	223	304	304	304	304	304	304	1.62	1.62	1.62	1.62
	238	342	342	342	342	342	342	1.72	1.72	1.72	1.72
	275	302	318	318	302	318	318	0	0	0	0
6 × 6	312	324	339	339	324	339	339	2.59	2.59	2.59	2.59
	331	324	339	339	324	339	339	2.59	2.59	2.59	2.59
	375	329	336	336	336	336	336	0	0	0	0
7 × 7	426	336	349	350	349	350	350	0.28	0	0	0
	453	336	374	374	361	374	374	0	0	0	0
	497	297	297	297	297	297	297	3.88	3.88	3.88	3.88
8 × 8	562	297	297	297	297	297	297	3.88	3.88	3.88	3.88
	593	297	297	297	297	297	297	3.88	3.88	3.88	3.88

Table 6.25. Results for LS and TS with DH at high energy level (cont)

		Lower Bound						Accuracy			
		S = 2			S = 3			S = 2		S = 3	
n × n	B	DH	LS_{DH}	TS_{DH}	DH	LS_{DH}	TS_{DH}	LS_{DH}	TS_{DH}	LS_{DH}	TS_{DH}
	622	308	330	330	330	330	330	0	0	0	0
9 × 9	705	310	338	338	331	338	338	0	0	0	0
	745	310	346	346	331	346	346	0	0	0	0
	754	327	327	327	327	327	327	6.03	6.03	6.03	6.03
10 × 10	855	337	364	364	337	367	367	5.70	5.70	5.70	5.70
	903	337	400	400	337	400	400	0	0	0	0
	1686	315	321	321	315	—	—	0	0	—	—
15 × 15	1918	342	342	342	342	—	—	10.94	10.94	—	—
	2024	342	342	342	342	—	—	10.94	10.94	—	—
	2958	292	—	—	292	—	—	—	—	—	—
20 × 20	3369	343	—	—	384	—	—	—	—	—	—
	3564	343	—	—	384	—	—	—	—	—	—
Average:		317.0	330.0	330.1	324.1	327.0	328.7	2.32	2.31	2.41	1.88
Adjusted Avg:		313.4	329.3	329.4	318.6	327.0	328.7	1.69	1.68	2.41	1.88

On the other hand, even after calculating the average lifetimes for the same number of instances, both of the search algorithms cannot improve the average lifetime with the increasing number of sinks in the network. This is again a result of the weak performance of the algorithms in 4×4 with low budget instance.

The efficiencies of the search algorithms can be evaluated with the accuracy values which are also given in the above tables. According to the adjusted accuracies, the search algorithms with two sinks are more efficient than the ones with three sinks at all energy levels. This result is due to the poor performance of the search algorithms in 4×4 instances. Besides, given the number of sinks in the network, algorithm TS is more accurate than algorithm LS at all energy levels. Therefore, we can say that algorithm TS is more reliable than algorithm LS for both number of sinks at all energy levels.

The computational times of the search algorithms are reported in Table 6.26 - Table 6.28. Comparing the complexities of the search algorithms, we expect to have higher computational times for algorithm TS than algorithm LS and for a search algorithm with three sinks than a search algorithm with two sinks. Notice that the results in the below tables are contradicting with the expectation. First, the running times for algorithm LS come out to be higher than the ones for algorithm TS due to the overload in the computer that these runs are obtained. Second, we observe that the results agree with our expectation related with the effect of sink number on computational time when we compare the average CPU times for the same number of instances.

Table 6.26. CPU times for LS and TS with DH at low energy level

		CPU Times (s)					
		S = 2			S = 3		
n × n	B	DH	LS_{DH}	TS_{DH}	DH	LS_{DH}	TS_{DH}
	111	0.99	27.79	20.52	1.03	48.66	26.14
4 × 4	126	1.10	32.61	24.19	1.19	39.98	44.36
	136	1.18	32.54	22.90	1.16	60.56	46.63
	197	1.49	44.14	38.98	1.62	87.73	64.74
5 × 5	223	1.50	44.23	39.61	1.65	87.82	74.64
	238	1.59	46.98	44.12	1.80	107.11	81.00
	275	2.38	94.59	80.16	2.45	232.07	132.49
6 × 6	312	2.53	99.74	84.10	2.53	247.14	135.59
	331	2.41	105.81	85.97	2.54	232.98	116.75
	375	3.02	188.71	146.56	3.87	389.83	256.64
7 × 7	426	3.11	237.18	161.68	4.01	291.92	287.30
	453	3.21	242.61	180.96	4.07	348.46	293.52
	497	4.47	359.64	220.98	5.62	508.52	326.81
8 × 8	562	4.51	336.16	220.42	5.90	581.45	282.25
	593	4.82	361.51	216.19	5.57	557.26	318.15
	622	7.37	738.95	413.96	9.77	1542.81	798.06
9 × 9	705	8.47	823.71	408.10	10.21	1434.50	924.14
	745	7.73	797.61	436.62	9.92	2305.93	855.01
	754	11.99	967.55	658.36	14.52	1333.40	925.05
10 × 10	855	11.87	1362.61	996.70	15.55	1662.33	1609.94
	903	12.07	1292.84	917.08	15.45	2256.72	2112.48
	1686	95.97	7388.84	3406.78	79.38	—	—
15 × 15	1918	98.60	9814.85	4828.71	92.99	—	—
	2024	94.23	9861.60	4412.94	93.78	—	—
	2958	247.74	—	—	231.26	—	—
20 × 20	3369	270.80	—	—	253.48	—	—
	3564	89.78	—	—	260.01	—	—
Average:		36.85	1470.95	752.77	41.90	683.68	462.46
Adjusted Avg:		4.66	392.26	258.01	5.74	683.68	462.46

Table 6.27. CPU times for LS and TS with DH at medium energy level

		CPU Times (s)					
		S = 2			S = 3		
n × n	B	DH	LS_{DH}	TS_{DH}	DH	LS_{DH}	TS_{DH}
	111	1.50	59.33	36.30	1.49	104.40	66.82
4 × 4	126	1.59	57.91	47.08	1.68	125.63	81.29
	136	1.65	67.79	41.16	1.70	113.76	55.82
	197	2.77	77.15	79.36	2.54	159.02	121.66
5 × 5	223	2.23	84.54	77.52	2.57	189.39	101.84
	238	2.25	95.78	85.38	2.87	164.66	131.94
	275	3.52	179.88	162.69	4.27	439.57	277.85
6 × 6	312	3.22	265.01	174.34	4.32	366.25	264.45
	331	3.26	235.27	170.66	4.35	413.08	279.19
	375	5.46	442.34	294.62	7.20	669.73	525.08
7 × 7	426	5.44	476.33	351.20	7.25	675.96	529.21
	453	5.57	429.33	335.46	7.44	868.93	595.42
	497	8.22	758.24	429.80	10.35	1007.91	618.18
8 × 8	562	8.30	727.65	444.03	10.34	1949.39	599.21
	593	8.20	790.66	475.69	10.32	1819.90	567.36
	622	14.14	1159.56	835.38	19.06	1930.70	1372.07
9 × 9	705	13.93	1225.37	946.36	18.38	4004.39	1535.05
	745	14.01	1254.97	1370.47	18.96	2915.12	1423.78
	754	22.85	2272.38	2162.63	28.48	4093.26	1791.68
10 × 10	855	25.53	2181.28	1901.90	28.90	4428.63	3233.14
	903	28.74	1718.14	2678.74	29.48	4828.36	2878.07
	1686	149.50	19097.13	8080.22	158.90	—	—
15 × 15	1918	171.17	21255.69	8740.21	165.87	—	—
	2024	166.84	21282.05	8295.87	175.80	—	—
	2958	418.04	—	—	824.48	—	—
20 × 20	3369	514.98	—	—	590.98	—	—
	3564	526.29	—	—	592.88	—	—
Average:		78.84	3174.74	1592.38	101.14	1488.95	811.86
Adjusted Avg:		8.66	693.28	623.85	10.57	1488.95	811.86

Table 6.28. CPU times for LS and TS with DH at high energy level

		CPU Times (s)					
		S = 2			S = 3		
n × n	B	DH	LS_{DH}	TS_{DH}	DH	LS_{DH}	TS_{DH}
	111	1.58	72.59	50.38	1.71	146.08	76.98
4 × 4	126	2.13	60.46	58.25	2.14	109.31	94.68
	136	1.93	67.78	59.69	1.97	134.33	91.00
	197	2.78	128.24	113.26	3.28	303.43	182.23
5 × 5	223	3.36	135.18	106.74	3.44	272.47	179.26
	238	3.84	138.92	119.01	3.65	320.56	201.82
	275	4.18	382.41	230.24	5.71	584.07	408.16
6 × 6	312	4.46	363.59	246.91	6.13	666.79	381.91
	331	4.50	396.32	245.73	5.72	614.87	416.44
	375	7.72	650.36	466.67	10.02	981.59	607.09
7 × 7	426	7.96	640.65	470.92	10.42	908.64	631.93
	453	7.92	686.39	499.65	10.93	1141.01	628.59
	497	12.15	848.51	648.58	14.84	1486.47	864.11
8 × 8	562	12.01	914.77	658.54	15.10	2889.28	827.58
	593	12.33	1033.17	676.32	15.11	2249.43	801.75
	622	20.48	2123.50	1180.78	27.59	3109.71	2098.05
9 × 9	705	20.40	2087.53	1382.94	27.46	2555.94	2249.59
	745	20.76	2052.93	1247.09	28.13	4262.53	1550.81
	754	42.00	2869.12	1746.05	37.56	4105.32	2130.09
10 × 10	855	40.32	4881.37	3022.62	39.06	7776.44	5399.86
	903	47.06	6673.36	3242.76	36.84	6309.54	4472.18
	1686	228.86	26366.94	12082.11	249.20	—	—
15 × 15	1918	234.07	26116.28	14096.34	255.13	—	—
	2024	254.20	26431.46	11882.20	240.53	—	—
	2958	654.14	—	—	864.22	—	—
20 × 20	3369	697.69	—	—	1369.46	—	—
	3564	697.16	—	—	1365.28	—	—
Average:		112.82	4421.74	2272.24	172.25	1948.94	1156.86
Adjusted Avg:		13.33	1295.58	784.44	14.61	1948.94	1156.86

The response of the search algorithms to the number of sinks and the subalgorithms, i.e. GH or DH, are illustrated in Figure 6.10 and Figure 6.11, respectively. The search algorithms are also compared in Figure 6.12 with each other with their best performing subalgorithms in terms of network lifetime. The computational times for the search algorithms with two and three sinks in the network are given in Figure 6.13 and Figure 6.14, respectively. The indifference of the lifetimes, that are obtained by the search algorithms with subalgorithm GH, when the number of sinks in the network is increased from two to three can be seen in Figure 6.10. On the other hand, subalgorithm DH can enhance the network lifetime as the number of sinks in the network increases only at low energy level for each of the search algorithms. The best performing subalgorithm for a search algorithm can be found by analyzing the Figure 6.11. At all energy levels and for both number of sinks, subalgorithm DH works better than subalgorithm GH for both of the search algorithms. Therefore, we can conclude that for both of the search algorithms DH is the best performing subalgorithm under both number of sinks.

The search algorithms LS and TS are compared under different number of sinks with their best performing subalgorithms, i.e. subalgorithm DH for both of the algorithms as we observe in Figure 6.11. For both number of sinks in the network and at all energy levels algorithm TS gives better lifetimes than algorithm LS.

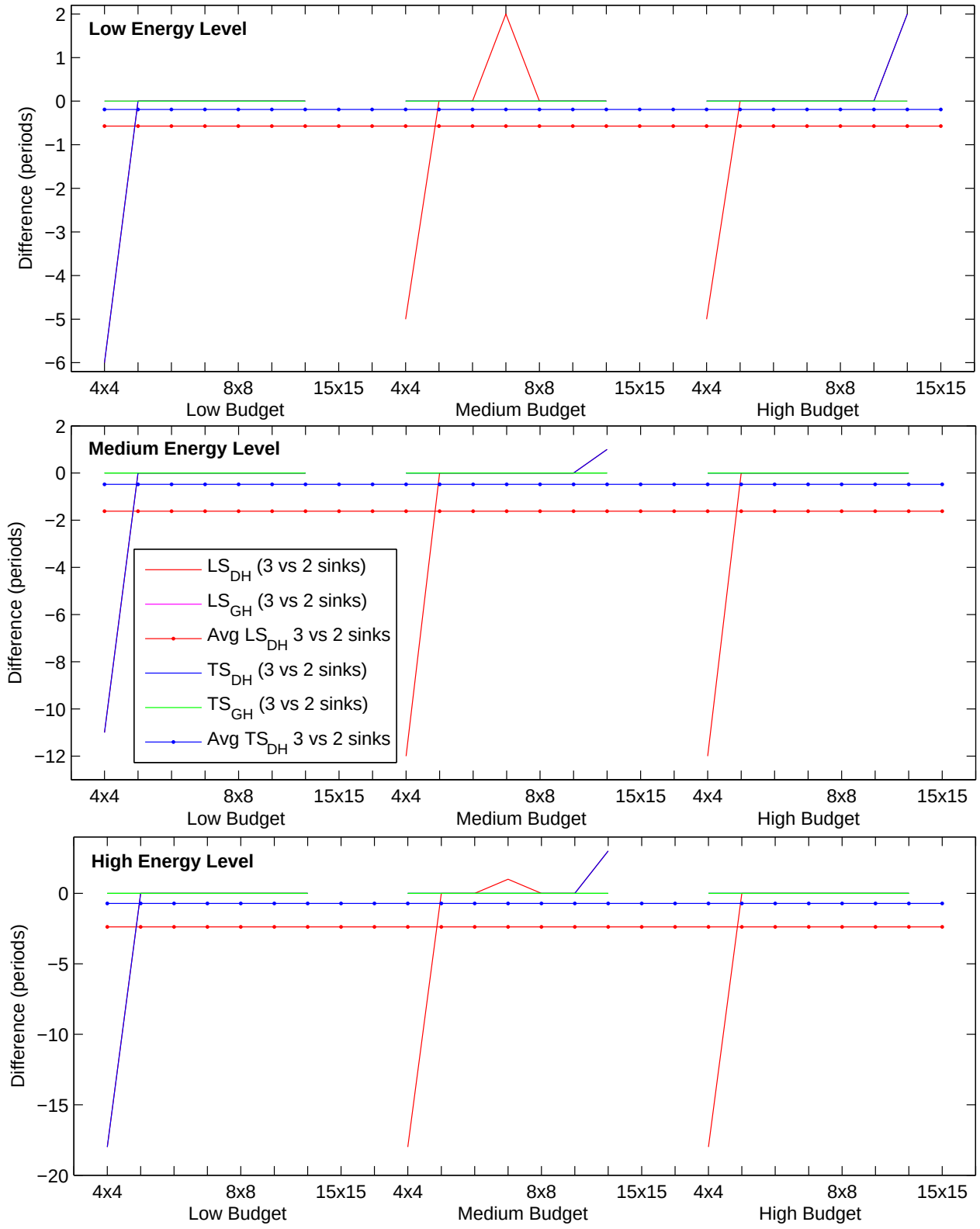

Figure 6.10. Sensitivity of the algorithms LS and TS to the number of sinks

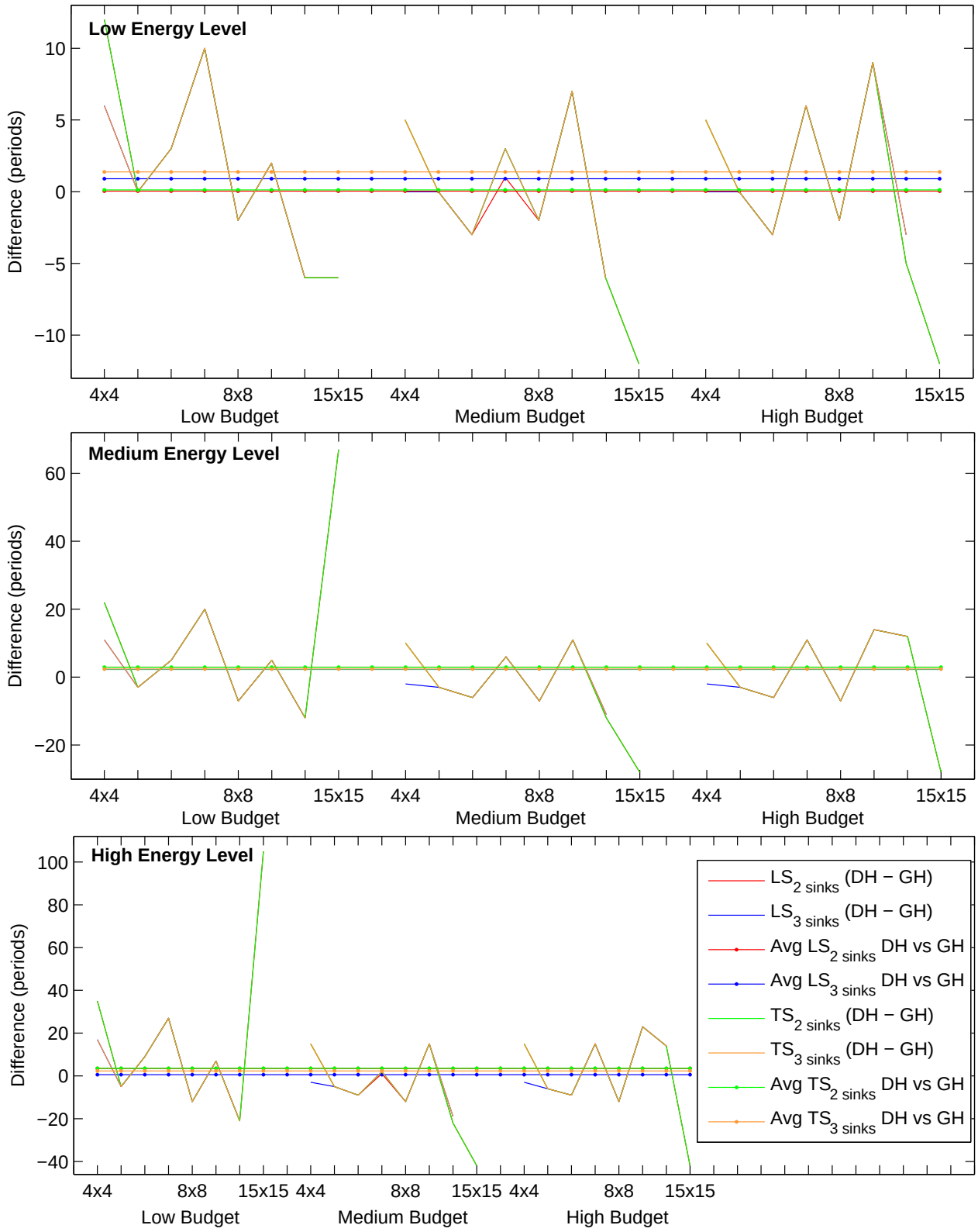

Figure 6.11. Performance of the algorithms LS and TS with GH and DH

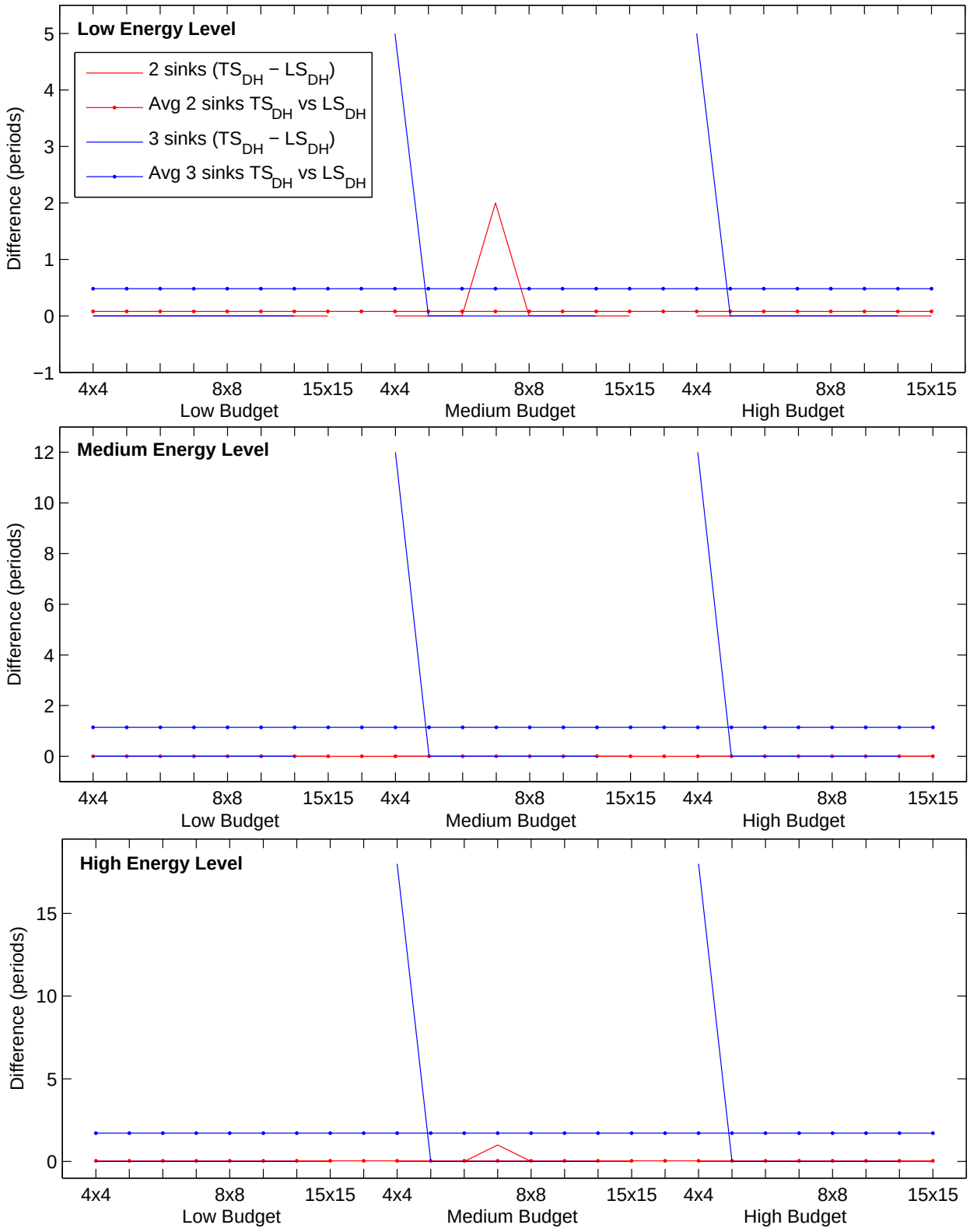

Figure 6.12. Comparison of the algorithms LS and TS with their best subalgorithms

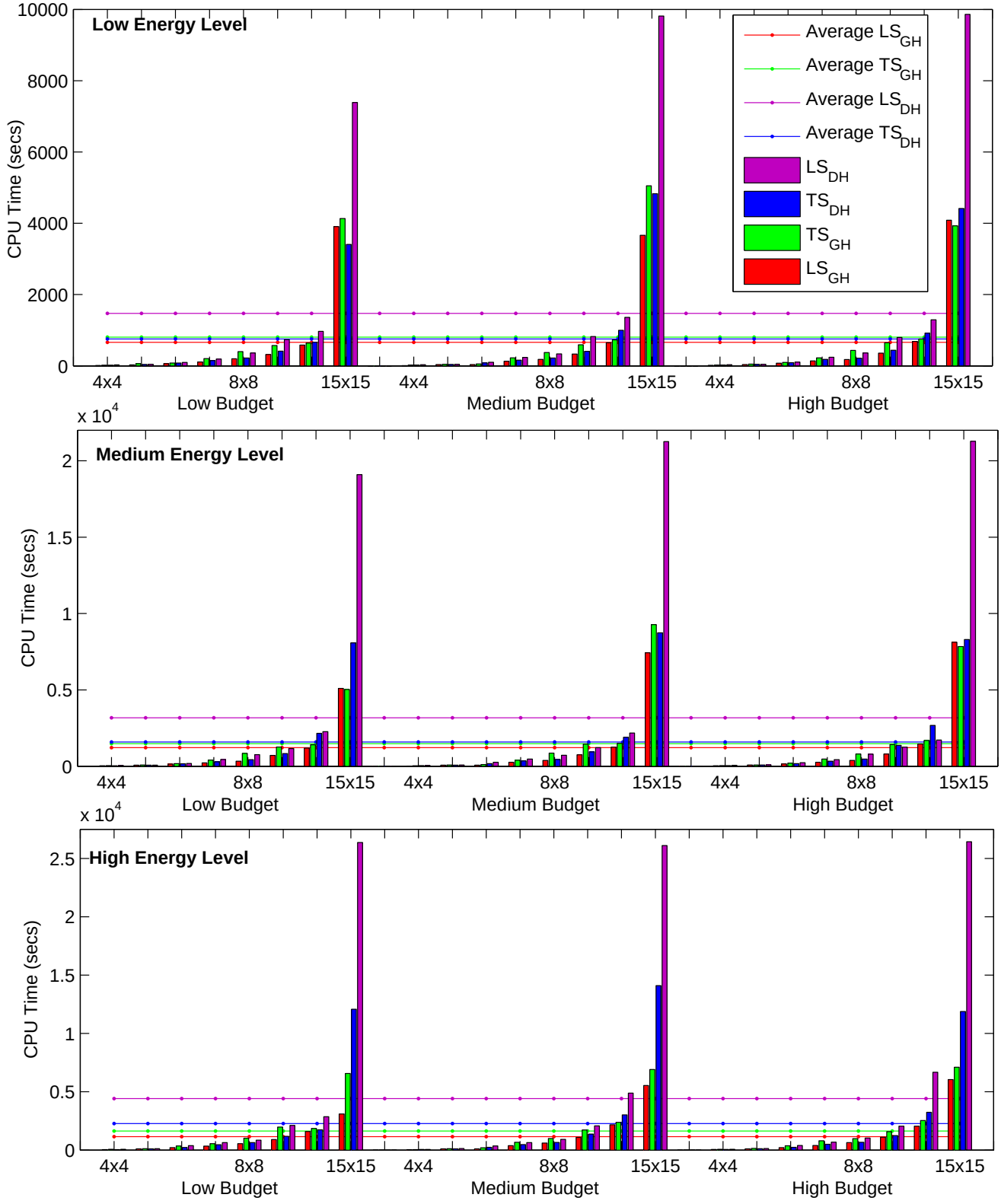

Figure 6.13. CPU times for LS and TS algorithms with 2 sinks

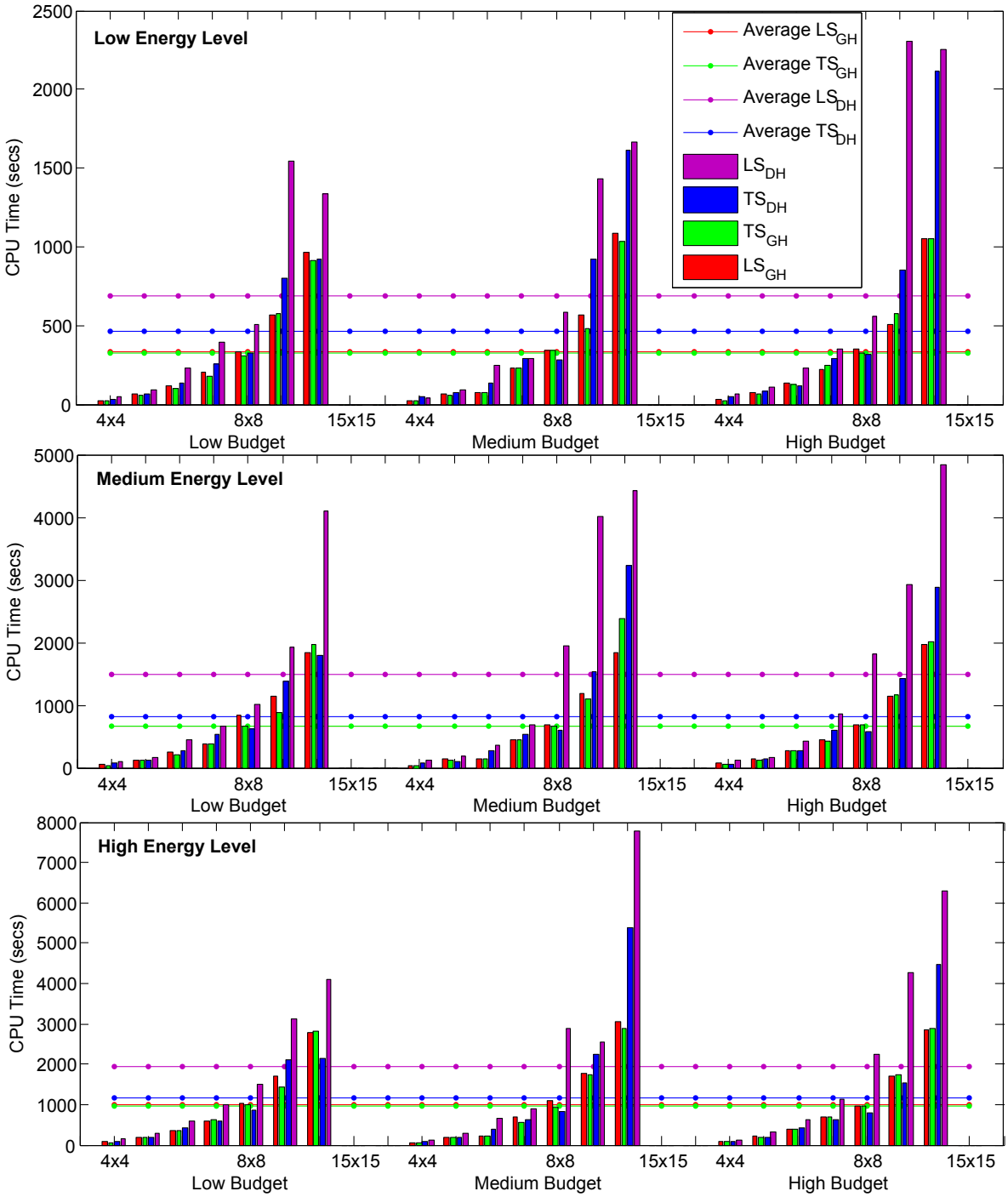

Figure 6.14. CPU times for LS and TS algorithms with 3 sinks

7. CONCLUSIONS

In this thesis we first worked on Sensor Placement, Scheduling and Routing Problem with Connectivity Restrictions (PSRPC). We formulated a MILP for the problem and proposed two Lagrangean Heuristics that make use of different relaxation strategies for the solution procedure. Experiments show that the Lagrangean Heuristics are performing better than the commercial solver for L in three hours. They generate a feasible solution as lower bound and an upper bound for the large instances of the problem for which the commercial solver can exibit neither a feasible solution nor an upper bound. Besides, the first Lagrangean Heuristic outperforms than the second one in terms of the quality of the upper bound and the computation time. We also introduced new valid inequalities for the PSRPC in addition to the valid inequalities proposed in the literature.

We then considered the problem of locating sinks integrated to the PSRPC. A MILP formulation is given for the new problem. Two different search algorithms, namely Local Search and Tabu Search, are developed with Greedy Heuristic or Discrimination Heuristic, the feasible solution generating heuristics used in the Lagrangean Heuristics for PSRPC. The algorithms search for a set of good locations for the sinks to maximize the network lifetime. The results show that for both of the search algorithms DH gives better lower bounds for the network lifetime than GH. Moreover, the search algorithms can improve the lower bound when the number of sinks in the sensor field increases with algorithm DH.

In this thesis experiments are conducted in order to determine the performance of the feasible solution generation heuristics under three different energy and three different budget levels. The results indicate that Discrimination Heuristic gives better lower bounds than Greedy Heuristic while consuming more time. Moreover, the effect of number of sinks on the network lifetime is investigated with both of the heuristics. We observed that the network lifetime increases significantly as the number of sinks in the sensor field increases. From the computational results we observe that the limited

resources budget and energy are consumed for providing coverage and routing the data to the sink nodes. Therefore to maximize the lifetime of a WSN one should consider efficient coverage and communication strategies together.

As a future research one can introduce alternative formulations for PSRPC from which a polynomial time algorithm can be obtained through Lagrangean relaxation or some other technique which can be based on column generation or branch and price methods. Alternative feasible solution generation algorithms can be developed which improve the lower bound. In this study the sensor field was consisting of discrete points. It may possible to consider coverage of a continuos region, means sensor field includes infinitely many points, where the coverage quality of a sensor decays as the Euclidean distance increases. Another research can be on developing algorithms for activity scheduling of the sensors and data routing without the information of sensor locations for large size sensor networks. We have assumed that sinks are stable during the network lifetime. Therefore, a future study can focus on the mobile sinks which can help to decrease the energy consumption in data transmission. Moreover, assuming probabilistic communication case and imposing bounds on the data flows among sensors can extend the topic of this thesis.

REFERENCES

Akyildiz, I. F., W. Su, Y. Sankarasubramaniam and E. Çayırcı, 2002, "Wireless sensor networks: a survey", *Computer Networks*, Vol. 38, pp. 393-422.

Altınel, İ. K., N. Aras, E. Güney and C. Ersoy, 2008, "Binary integer programming formulation and heuristics for differentiated coverage in heterogeneous sensor networks", *Computer Networks*, Vol. 52, Iss. 12, pp. 2419-2431.

Arai, S., Y. Iwatani and K. Hashimoto, 2010, "Fast sensor scheduling with communication costs for sensor networks", *Proceedings of American Control Conference*, ACC, Baltimore, MD, pp. 300.

Arya, V., N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit, 2004, "Local search heuristics for k-median and facility location problems", *SIAM Journal on Computing*, Vol. 33, No. 3, pp. 544-562.

Beasley, J. E. 1993, "Lagrangean heuristics for location problems", *European Journal of Operational Research*, Vol. 65, Iss. 3, pp. 383-399.

Byun, T. Y., M. Kim, S. Hwang and S. E. Jeon, 2006, "An active node set maintenance scheme for distributed sensor networks", *Lecture Notes in Computer Science*, Springer Verlag, Berlin, Vol. 3982, pp. 134-143.

Cardei, M. and J. Wu, 2006, *Coverage in wireless sensor networks*, Technical Report, Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, Florida.

Chakrabarty, K., S. S. Iyengar, H. Qi and E. Cho, 2002, "Grid coverage and surveillance and target location in distributed sensor networks", *IEEE Transactions on Computers*, Vol. 51, Iss. 12, pp. 1448.

Chaudhry, S. B., V. C. Hung, R. K. Guha and K. O. Stanley, 2010, "Pareto-based evolutionary computational approach for wireless sensor placement", *Engineering Applications of Artificial Intelligence*, doi:10.1016/j.engappai.2010.07.007.

Cheng, X., D. Z. Du, L. Wang and B. Xu, 2008, "Relay sensor placement in wireless sensor networks", *Wireless Networks*, Vol. 14, pp. 347-355.

Dhillon, S. S. and K. Chakrabarty, 2003, "Sensor placement for effective coverage and surveillance in distributed sensor networks", *Wireless Communications and Networking*, Vol. 3, pp. 1614.

Ergen, S. C. and P. Varaiya, 2006, "Optimal placement of relay nodes for energy efficiency in sensor networks", *Proceedings of IEEE International Conference on Communications*, IEEE Communications Society, Piscataway, NJ, pp. 3473-3479.

Fei, X., S. Samarah and A. Boukerche, 2010, "A bio-inspired coverage-aware scheduling scheme for wireless sensor networks", *Proceedings of IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum*, IEEE Computer Society, Piscataway, NJ, pp. 8.

Gendreau, M. and J. Y. Potvin, 2005, "Metaheuristics in combinatorial optimization", *Annals of Operations Research*, Vol. 140, pp. 189-213.

Güney, E., N. Aras, İ. K. Altınel and C. Ersoy, 2010, "Efficient integer programming formulations for optimum sink location and routing in heterogeneous wireless sensor networks", *Computer Networks*, Vol. 54, pp. 1805-1822.

Held, M., P. Wolfe and H. P. Crowder, 1974, "Validation of subgradient optimization", *Mathematical Programming*, Vol. 6, pp. 62-88.

Hua, C. and T. S. P. Yum, 2008, "Optimal routing and data aggregation for maximizing lifetime of wireless sensor networks", *IEEE/ACM Transactions on Networking*, Vol. 16, No. 4, pp. 892-903.

Ilog, 2007, *ILOG CPLEX 11.0 User's Manual*, ILOG Inc., viewed 21 September 2010, <http://www.lingnan.net/lab/uploadfile/200864184419679.pdf>.

Karl, H. and A. Willig, 2003, *A short survey of wireless sensor networks*, TKN Technical Report TKN-03-018, Technical Univerity Berlin.

Karmarkar, N. 1984, "A new polynomial-time algorithm for linear programming", *Combinatorica*, Vol. 4, pp. 373-395.

Li, Y. and S. Gao, 2008, "Designing k-coverage schedules in wireless sensor networks", *Journal of Combinatorial Optimization*, Vol. 15, pp. 127-146.

Lin, F. Y. S. and P. L. Chiu, 2005, "A near-optimal sensor placement algorithm to achieve complete coverage/discrimination in sensor networks", *IEEE Communications Letters*, Vol. 9, No. 1, pp. 43-46.

Lin, Y., X. Hu and J. Zhang, 2010, "An ant-colony-system-based activity scheduling method for the lifetime maximization of heterogeneous wireless sensor networks", *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference*, ACM, New York, pp. 233-240.

Liu, X., L. Huang, W. Shi and H. Xu, 2008, "Energy-efficient location-independent k-connected scheme in wireless sensor networks", in Ali Miri (ed.), *Wireless Sensor and Actor Networks II*, Springer, Boston, pp. 257-268.

Liu, Y. and W. Liang, 2008, "Prolonging network lifetime for target coverage in sensor networks", *Lecture Notes in Computer Science*, Springer Verlag, Berlin, Vol. 5258, pp. 212-223.

Liu, Z. and W. Xu, 2010, "Zeroing-in on network metric minima for sink location determination ", *Proceedings of the third ACM conference on Wireless network security*, ACM, New York, pp. 94-104.

Oyman, E. I. and C. Ersoy, 2004, "Multiple sink network design problem in large

scale wireless sensor networks", *Proceedings of IEEE International Conference on Communications*, IEEE Communications Society, Piscataway, NJ, Vol. 6, pp. 3663-3667.

Pazand, B. and A. Datta, 2006, "Minimum dominating sets for solving the coverage problem in wireless sensor networks", *Lecture Notes in Computer Science*, Vol. 4239, pp. 456-466.

Poe, W. Y. and J. B. Schmitt, 2009, "Sink placement without location information in large-scale wireless sensor networks", *Proceedings of Asian Internet Engineering Conference*, ACM, New York, pp. 69-76.

Sacchi, L. H. and V. A. Armentano, 2010, "A computational study of parametric tabu search for 0-1 mixed integer programs", *Computers & Operations Research*, Vol. 38, pp. 464-473.

Sahni, S. and X. Xu, 2005, "Algorithms for wireless sensor networks", *International Journal of Distributed Sensor Networks*, Vol. 1, Iss. 1, pp. 35-56.

Schurgers, C. and M. B. Srivastava, 2001, "Energy efficient routing in wireless sensor networks", *Communications for Network-Centric Operations: Creating the Information Force*, IEEE Military Communications Conference, Los Angeles, CA, Vol. 1, pp. 357-361.

Torres, M. G. and J. Kabara, 2006, "Measuring energy consumption in wireless sensor networks using GSP", *Proceedings of the 17th International Symposium on Personal, Indoor and Mobile Radio Communications*, IEEE, Piscataway, NJ, pp. 1-5.

Türkoğulları, Y. B., N. Aras, İ. K. Altınel and C. Ersoy, 2010a,"An efficient heuristic for placement, scheduling and routing in wireless sensor networks", *Ad Hoc Networks*, Vol. 8, pp. 654-667.

Türkoğulları, Y. B., N. Aras, İ. K. Altınel and C. Ersoy, 2010b,"A column generation

based heuristic for sensor placement, activity scheduling and data routing in wireless sensor networks", *European Journal of Operational Research*, Vol. 207, pp. 1014-1026.

Türkoğulları, Y. B. 2010c, "Optimal placement, scheduling and routing to maximize lifetime in wireless sensor networks", PhD thesis, Boğaziçi University, İstanbul.

Wang, B., K. C. Chua, V. Srinivasan and W. Wang, 2009, "Scheduling sensor activity for information coverage of discrete targets in sensor networks", *Wireless Communications and Mobile Computing*, Vol. 9, pp. 745-757.

Wang, L. and Y. Xiao, 2006, "A survey of energy-efficient scheduling mechanisms in sensor networks", *Mobile Networks and Applications*, Vol. 11, pp. 723-740.

Wang, J. and N. Zhong, 2006, "Efficient point coverage in wireless sensor networks", *Journal of Combinatorial Optimization*, Vol. 11, pp. 291-304.

Xbow, 2009, *440 Series User's Manual*, Crossbow Technology Inc., viewed 21 September 2010, <http://www.xbow.com/pdf/440_Series_Inertial_Manual.pdf>.

Yang, L. 2006, "Determining sink node locations in wireless sensor networks", *Proceedings of IEEE INternational Conference on Systems, Man and Cybernetics*, IEEE, Piscataway, NJ, pp. 3400-3404.

Yang, S., F. Dai, M. Cardei, J. Wu and F. Patterson, 2006, "On connected multiple point coverage in wireless sensor networks", *International Journal of Wireless Information Networks*, Vol. 13, No. 4, pp. 289-301.

Yang, Y. and M. Cardei, 2010, "Adaptive energy efficient sensor scheduling for wireless sensor networks", *Optimization Letters*, Vol. 4, pp. 359-369.

Yardibi, T. and E. Karasan, 2010, "A distributed activity scheduling algorithm for wireless sensor networks with partial coverage", *Wireless Networks*, Vol. 16, pp. 213-225.

Ye, Y. 1991, "An $\mathcal{O}(n^3L)$ potential reduction algorithm for linear programming", *Mathematical Programming*, Vol. 50, pp. 239-258.

Yuan, Z., R. Tan, G. Xing, C. Lu, Y. Chen and J. Wang, 2008, "Fast sensor placement algorithms for fusion-based target detection", *Proceedings of the Real Time Systems Symposium*, IEEE Computer Society, Piscataway, NJ, pp. 103-112.