

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

by

Yavuz Boğaç Türkoğulları

B.S., Industrial Engineering, Boğaziçi University, 2001

M.S., Industrial Engineering, Galatasaray University, 2004

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

Graduate Program in Industrial Engineering
Boğaziçi University

2010

ACKNOWLEDGEMENTS

I would like to thank my cosupervisors Prof. Dr. İ. Kuban Altınel and Assoc. Prof. Necati Aras for their guidance, encouragement, and patience while completing this work. From the beginning with my Ph.D. degree study, I have learned lots of things from them. I think I am very lucky to have chance to study with them.

I am grateful to other members of my committee, Prof. Dr. Cem Ersoy, Prof. Dr. Taner Bilgiç and Ass. Prof. Ayşegül Gençata Yayımlı for their interest, comments, encouragement, and willingness to serve as my committee members.

I am very grateful to my family. My mother, Nurcan Türkoğulları, made me feel good in difficult times. My brother, Boğaç Türkoğulları, is the person from whom I learned how to handle daily life problems.

I gratefully acknowledge the support of TÜBİTAK (The Scientific and Technological Research Council of Turkey) under the grant no: 107M250 and Boğaziçi University Research Fund under the Grant no: 08A304D.

ABSTRACT

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

A wireless sensor network consists of distributed autonomous tiny electronic devices called sensors. They have limited energy and capability for sensing, data processing and communication, but they can collectively behave to provide an effective network that monitors a region, and transmit information to gateway nodes called sinks. In most of the applications, the network must operate for long periods of time, so the available energy resources of the sensors must be managed efficiently. In this thesis we first develop mixed-integer linear programming models to maximize network lifetime by optimally determining locations of sensors and sinks, sensor-to-sink data routes and activity schedules of the deployed sensors over a finite planning horizon subject to coverage, flow conservation, energy consumption and budget constraints. Then, we propose valid inequalities, which we use with a new branch-and-price algorithm to solve the problem exactly. Although they provide stronger linear programming relaxations and thus increase the quality of the formulation, the size of the networks for which the model can be solved exactly is very limited. Hence, we develop approximate solution heuristics using techniques such as the Lagrangean relaxation, column generation and a greedy selection criterion. Based on the computational experiments we can say that they are accurate and efficient.

The minimum cost coverage problem, which consists of the deployment of sensors over a sensor field modeled as a set of finite points so that total cost is minimized subject to the coverage requirements, generalizes the famous set covering problem. We provide a polyhedral analysis of this general problem, which has not been studied very much previously. We develop a procedure that generates valid inequalities and

a sufficient condition for facet definition, and a cutting plane. Experiments show that the proposed valid inequality cuts of the fractional optimum and improve considerably the linear programming relaxation lowerbound.

Finally, we revise the convergence proof of the deflected subgradient algorithm that uses conditional subgradients and suggest some refinements.

ÖZET

KABLOSUZ DUYGAÇ AĞLARINDA AĞ ÖMRÜNÜ EN BÜYÜKLEMEK İÇİN ENİYİ YERLEŞTİRME, ÇİZELGELEME VE ROTALAMA

Bir kablosuz duygaç ağı, duygaç adı verilen, dağıtık ve bağımsız çalışabilen minik elektronik aygıtlardan oluşur. Duygaçlar duyumsama, veri işleme ve iletme için sınırlı enerji kaynakları, yetenekleri olmasına karşın, birlikte davranarak bir alanı gözetleyen, toplanan bilgiyi ana alıcılara ileten etkin bir ağ oluşturabilirler. Uygulamaların çoğunda ağın uzun süre yaşamasının zorunlu olması nedeniyle duygaçların bireysel enerji kaynaklarının çok iyi yönetilmesi gerekmektedir. Bu konuya odaklanan bu tezde, ilkin duygaçlarla alıcıların eniyi yerlerini, duygaçlardan alıcılara olan eniyi bilgi akış rotalarını ve eniyi etkinlik çizelgelerini bularak ağın ömrünü enbüyükleyen karışık tamsayılı programlama gösterimleri geliştirilmektedir. Gösterimler, duygaçların ve alıcıların eniyi yerlerini, duygaçların açık/kapalı oldukları çevrimleri ve duygaçlardan alıcılara olan veri akış rotalarını, ağ ömrünü verilmiş sonlu bir planlama çevreni içinde enbüyükleyecek biçimde bulabilirler. Daha sonra, yeni bir dal-ve-eder algoritmasıyla birleştirilen yeni geçerli eşitsizlikler önerilmektedir. Her ne kadar bunları kullanmak daha güçlü doğrusal programlama gevşetmelerine neden olarak niteliklerini artırsa da, gösterimlerin eniyi çözümlerinin hesaplanabileceği ağların boyutları çok sınırlıdır. Bu nedenle, yaklaşık çözümler bulan, Lagrange gevşetmesi, sütun üretme ve açgözlü seçim yöntemlerini kullanan sezgiseller geliştirilmektedir. Yapılan bilgisayarlı deneyler bunların etkin ve doğru olduklarını söyler.

Duygaçları, belirlenen bir alanı kaplayacak ve toplam maliyeti enküçükleyecek biçimde yerleştirilmelerinden oluşan endüşük maliyetli kaplama problemi, Enküçük Ağırlıklı Küme Kaplama Problemi'ni geneller. Daha önce çok incelenmemiş olan bu genel problem, bir temel araştırma olarak bu tezin içerisinde ele alınmakta, tamsayı

programının olurlu çözüm kümesinin dışbükey örtüsü için yeni bir yüzlem ile kesik kaldırma yöntemi kullanılarak elde edilmektedir. Bunlar genelleştirilmiş problemin ikil tamsayı program gösterimini güçlendirerek daha iyi doğrusal programlama gevşetmesi altsınırları elde edilmesini sağlarlar ve doğrusal programlama gevşetmesinin kesirli çözümlerini keserler.

Tezin kapsamında son olarak saptırılmış ve koşullu altgradyan algoritmasının yakınsama tanıtı da gözden geçirilmekte ve gerekli görünen düzenlemeler yapılmaktadır.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	vi
LIST OF FIGURES	xi
LIST OF TABLES	xiv
LIST OF SYMBOLS/ABBREVIATIONS	xvi
1. INTRODUCTION	1
2. LITTERATURE SURVEY	5
2.1. Introduction	5
2.2. Sensor Placement	5
2.3. Sensor Activity Scheduling	7
2.4. Data Routing	10
2.5. Sink Location	14
2.6. Works That Integrate Some of the Design Issues	15
3. LOCATION AND SCHEDULING PROBLEM	18
3.1. Introduction	18
3.2. Model Formulation	18
3.3. Solution Procedure	20
4. A NEW VALID INEQUALITY AND CUTTING PLANE FOR COVERING BINARY INTEGER PROGRAMS	27
4.1. Introduction	27
4.2. A Valid Inequality for the Binary Covering Polytope	29
4.3. Sequential Lifting	31
4.4. Numerical Example: Valid Inequality	35
4.5. A Cutting Plane for the Binary Covering Integer Programs	38
4.6. Numerical Example: Cutting Plane	46
5. LOCATION, SCHEDULING AND ROUTING PROBLEM	49
5.1. Introduction	49
5.2. Model Formulation	49

5.3. Valid Inequalities	55
6. A BRANCH AND PRICE ALGORITHM	60
6.1. Introduction	60
6.2. Reformulation	60
6.3. Solution Procedure	64
6.3.1. Solution of the Reformulation	64
6.3.2. A Branch and Price Algorithm	68
7. A LAGRANGEAN HEURISTIC	77
7.1. Introduction	77
7.2. Lagrangean Heuristic	77
7.2.1. First Subproblem	78
7.2.2. Second Subproblem	79
7.2.3. Third Subproblem	81
7.2.4. Fourth Subproblem	82
7.2.5. Generating a Feasible Solution	83
8. A COLUMN GENERATION BASED HEURISTIC	95
8.1. Introduction	95
8.2. Determination of Sink Locations	95
8.3. Reformulation	96
8.4. Solving the LP Relaxation of the Reformulation Using Column Generation	101
8.5. Construction of a Feasible Solution	105
8.6. Implementation Issues	108
9. DISJOINT SETS HEURISTIC	110
9.1. Introduction	110
9.2. Disjoint Sets Heuristic	110
10. COMPUTATIONAL STUDY	120
10.1. Introduction	120
10.2. Location and Scheduling Problem	121
10.3. Covering Integer Program	122
10.4. Location, Scheduling and Routing Problem	127
10.4.1. The Strength of the New Valid Inequality and Cutting Plane .	127
10.4.2. Quality of the Upper Bound	131

10.4.3. Accuracy and Efficiency of the New Heuristics	133
10.4.4. Experiments With Very Large Instances	138
10.4.4.1. The Performance of the Disjoint Sets Heuristic	139
10.4.4.2. The Effect of Data Packet Generation Rate on the Life- time	140
11. CONCLUSIONS	145
APPENDIX A: ON THE RELATION BETWEEN COVERAGE AND CONNEC- TIVITY	147
APPENDIX B: ON THE CONVERGENCE OF THE DEFLECTED SUBGRADI- ENT ALGORITHM	149
REFERENCES	169

LIST OF FIGURES

Figure 1.1.	A WSN which collects temperature and moisture data	2
Figure 3.1.	Solution algorithm for the first subproblem	22
Figure 3.2.	Solution algorithm for the second subproblem	23
Figure 3.3.	Lagrangean Heuristic	25
Figure 3.4.	Generation of a feasible solution	26
Figure 4.1.	Valid inequality generation	35
Figure 4.2.	Construction of a cutting plane	44
Figure 4.3.	Greedy heuristic	46
Figure 5.1.	A sample sensor network operating for $T = 2$ periods	55
Figure 6.1.	Solution of RLSRP (first part)	69
Figure 6.2.	Solution of RLSRP (second part)	70
Figure 6.3.	Solution of RLSRP (third part)	71
Figure 6.4.	Solution of RLSRP (fourth part)	72
Figure 6.5.	Algorithm dual heuristic	73
Figure 6.6.	Algorithm primal heuristic	74

Figure 6.7.	Branch and price (first part)	75
Figure 6.8.	Branch and price (second part)	76
Figure 7.1.	Lagrangian Heuristic (first part)	84
Figure 7.2.	Lagrangian Heuristic (second part)	85
Figure 7.3.	Generation of a feasible solution (first part)	85
Figure 7.4.	Generation of a feasible solution (second part)	86
Figure 7.5.	Restoring coverage and budget constraints feasibility (first part) .	87
Figure 7.6.	Restoring coverage and budget constraints feasibility (second part)	88
Figure 7.7.	Locating the sinks	89
Figure 7.8.	Restoring feasibility with respect to connectivity	90
Figure 8.1.	Algorithm location of sinks	96
Figure 8.2.	Solving the LP relaxation of RP by column generation	105
Figure 8.3.	Algorithm construction of a feasible solution to LSRP	108
Figure 9.1.	Algorithm disjoint set finder	112
Figure 9.2.	Algorithm enlargement of the disjoint set for connectivity	114
Figure 9.3.	Algorithm removing cycles	118

Figure 9.4. Algorithm disjoint sets heuristic	119
Figure 10.1. Lifetime as a function of the number of points.	142
Figure 10.2. CPU time as a function of the number of points.	142
Figure 10.3. Lifetime as a function of the data packet generation rate of sensors.	144
Figure A.1. A disconnected sensor placement that covers the sensor field	148

LIST OF TABLES

Table 5.1.	Decision variables used in the model	51
Table 5.2.	Parameters used in the model	52
Table 10.1.	Sensor specifications	120
Table 10.2.	Comparison of LH with CPLEX 11.0	123
Table 10.3.	Performance of LH on large LSP instances	124
Table 10.4.	Optimum values in the first set of test problems	125
Table 10.5.	Optimum values in the second set of test problems	126
Table 10.6.	Execution times for the first set of test problems	128
Table 10.7.	Execution times for the second set of test problems	129
Table 10.8.	The strength of the new valid inequalities	130
Table 10.9.	Comparison of the upper bounds for LSRP	132
Table 10.10.	Upper bounds for large LSRP instances	133
Table 10.11.	Lifetime values found by the methods proposed for LSRP	134
Table 10.12.	Accuracies of the methods proposed to solve LSRP	135
Table 10.13.	CPU Time consumed by the methods proposed to solve LSRP	136

Table 10.14. Lifetime values found by the methods proposed for LSRP on large instances	137
Table 10.15. Accuracies of the methods proposed to solve LSRP on large instances	138
Table 10.16. CPU Time consumed by the methods proposed to solve LSRP in large-sized instances	139
Table 10.17. Composition and active times of disjoint sets for the first set of experiments	141
Table 10.18. Composition and active times of disjoint sets for the second set of experiments	143
Table 10.19. Regression analysis for the relation between the CPU time of DSH and the number of points.	143

LIST OF SYMBOLS/ABBREVIATIONS

a_{ijk}	one if a type- k sensor located at point i can cover point j , zero otherwise
b_{ijk}	one if point j is within the transmission 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 at point j
d_j	coverage quality requirement for point j
e_{ik}^s	energy consumption of a type- k sensor located at point i for sensing and processing during a period
e_k^r	energy consumption of a type- k sensor for receiving a data packet
e_k^c	energy consumption of a type- k sensor for transmitting a data packet
E_k	initial battery energy of a type- k sensor
f_{ikjlt}	amount of data flow from a type- k sensor located at point i to a type- l sensor located at point j in period t
g_{ikjt}	amount of data flow from a type- k sensor located at point i to a sink placed at point j in period t
h_{ik}	number of data packets generated by a type- k sensor located at point i
L	lifetime of the WSN
r_k^s	sensing range of a type- k sensor
r_k^c	communication range of a type- k sensor
s_i	one if a sink is placed at point i , zero otherwise
S	number of sinks to be located
T	planning horizon
u_{kl}	upper bound on the amount of data sent from a type- k sensor to a type- l sensor
v_k	upper bound on the amount of data sent from a type- k sensor to a sink
x_{ik}	one if a type- k sensor is placed at point i , zero otherwise

y_t	one if period t is within the lifetime L , zero otherwise
z_{ikt}	one if a type- k sensor located at point i is active in period t , zero otherwise
WSN	wireless sensor network
CP	coverage problem
RP	routing problem
SLP	sink location problem
LSP	location and scheduling problem
MILP	mixed-integer linear program
LSRP	location, scheduling and routing problem
LP	linear program

1. INTRODUCTION

A *Wireless Sensor Network* (WSN), which consists of autonomous multi-functional, tiny electronic devices called *sensors*, is a network deployed over a designated region referred to as the *sensor field*. Sensors have limited energy and capability for sensing, data processing and communication, but they can collectively form a distributed network that provides a monitoring environment even for remote or inaccessible areas, which gives rise to the wide range of applicability of WSNs (Akyıldız et al. [1], Yick et al. [2]). Each sensor processes the data it collects within its *sensing range* to generate the information transmitted to gateway nodes called *sinks* either directly or through other sensors remaining within its *communication range*. Sensors can be of different types, and both their size and cost may vary depending on the size of the WSN and specifications of the sensors such as sensing and communication ranges, and initial battery energy. Figure 1.1 illustrates a WSN. Sensors shown by white circles collect temperature and moisture data at hundreds of points in the field. The sink is in the building. White lines show the data flow routes. Notice that sensors send the collected data either directly or indirectly through other sensors to the sink.

A typical WSN may consist of identical sensors giving rise to a *homogeneous* network, or may be composed of sensors with different technical characteristics resulting in a *heterogeneous* network. A sensor can be in either *active* or *standby* (i.e. inactive) mode. An active sensor performs its sensing, processing, data receiving, and information transmitting activities. A sensor in standby mode, on the other hand, does not perform any of these activities, and operates at the minimum energy level. The energy consumptions in sensing, data processing, message receiving and transmitting depend on the technology of the particular sensor type.

Because the sensors consume energy in sensing, processing, sending the collected data to the sink and their battery energy is limited, the WSN has a finite lifetime. In most WSN's it is impractical or costly to replace the batteries of the sensors because the WSN can be in a hostile or difficult to reach environment. In this thesis we maximize



Figure 1.1. A WSN which collects temperature and moisture data

the WSN's lifetime by determining optimally sensor and sink locations, sensor activity schedules and sensor-to-sink data flow routes.

One of the fundamental issues in the design of WSNs is to determine an effective sensor placement strategy to meet sensing quality requirements everywhere in the field with a certain flexibility of handling individual sensor failures. This is also known as the *Coverage Problem*(CP) (Cardei and Wu [3]). Sensing quality requirements, or simply *coverage requirements*, can be uniform or non-uniform over the sensor field. In the former case, we talk about the *uniform* CP while in the latter case we have a *differentiated* CP. An effective sensor placement strategy should also be *energy-aware*. It should decrease the energy consumption and increase the network lifetime. In the WSN of Figure 1.1 temperature and moisture data of all the points should reach the sink in the building. Locating the sensors in a way to provide this is within the scope of the sensor placement problem.

Apart from the placement of sensors, there is another way of obtaining energy

efficiency in WSNs: scheduling active and standby periods of deployed sensors (Wang and Xiao [4]). In a given time period only sensors necessary for the coverage of the sensor field and for the connectivity of the sensors are active, other sensors are in standby mode. Determining activity schedules of the sensors in a way to maximize the network lifetime is a crucial issue in the design of a WSN. In the WSN of Figure 1.1 all the sensors do not have to be active to collect temperature and moisture data if a subset of the sensors is sufficient for the coverage of the field. Determining the periods during which the sensors are active is within the scope of sensor activity scheduling problem.

Data transmission is another energy consuming sensor activity, which makes the determination of energy efficient sensor-to-sink data flow paths (also called routes) one of the fundamental issues in the design of WSNs (Akkaya and Younis [5], Oliveira and Pardalos [6]). When sensor and sink locations are known this is addressed as the *Routing Problem* (RP). In the WSN of Figure 1.1 we notice by following the white lines that some sensors send the collected temperature and moisture data directly to the sink in the building. However, some other sensors use their neighbors to send their data. Determining the data flow routes that minimize the energy consumption is within the scope of the RP.

Locating sink properly is another crucial issue in the design of a WSN since this decision also affects data routes, and thereby the energy spent on data transmission. Therefore, for minimizing the energy consumption or maximizing the network lifetime one should also pay attention to how to locate sinks. This problem, referred to as *Sink Location Problem* (SLP), is also studied by the researchers. For example, it is shown in Bogdanov et al. [7] that sink locations have an impact on the rate of data flow in WSNs, which ultimately affects the energy consumption.

In this thesis, we first concentrate on the *Location and Scheduling Problem* (LSP) which integrates sensor placement and sensor activity scheduling issues. The problem aims to determine the sensor locations and sensor activity schedules that maximize the network lifetime subject to coverage quality, energy and budget constraints. We

provide a mixed-integer linear programming (MILP) model for the LSP. We also devise a heuristic solution procedure in order to solve large LSP instances. The heuristic provides a feasible solution and an upper bound on the optimal lifetime value.

Then, we concentrate on the *Location, Scheduling, and Routing Problem* (LSRP) which integrates the main WSN design issues. It consists of the determination of the sensor and sink locations, sensor activity schedules and sensor-to-sink data flow routes that maximize the network lifetime subject to coverage requirements, energy, flow conservation and budget constraints. We provide a MILP model and devise an exact and several heuristic solution procedures in order to solve large LSRP instances. The heuristics provide a feasible solution (lower bound) and an upper bound on the optimal lifetime.

We also contribute to the mathematical programming literature by proposing a valid inequality and cutting plane for the general set covering polytope and revisiting the convergence proof for the convergence of the subgradient algorithm that uses both the deflected and the conditional subgradients.

In the remainder of the thesis we first review the literature related to the issues of sensor placement, sensor activity scheduling, data routing and sink location in the next chapter. We present a MILP formulation and a heuristic solution procedure for the LSP in the third chapter. In the fourth chapter a new valid inequality and cutting plane for the general set covering polytope are explained. In the fifth chapter the integrated MILP formulation for the LSRP is presented. In the sixth chapter a branch-and-price algorithm for the LSRP is proposed. In the seventh chapter a Lagrangean Heuristic is described for the LSRP which is then followed by a column generation based heuristic in the eighth chapter. A greedy heuristic is introduced in the ninth chapter. Tenth chapter is where the experimental results are reported. Finally the eleventh chapter concludes the thesis. In appendix A we give a counter example that shows that coverage does not always imply connectivity in an heterogenous sensor network. The new proof for the convergence of the subgradient algorithm that uses both the deflected and the conditional subgradients can be found in appendix B.

2. LITERATURE SURVEY

2.1. Introduction

In this chapter we briefly overview the works on the main WSN design issues: coverage, scheduling, routing and sink location problems. We first survey the ones dealing with only one of them. Then, we present integration attempts in order to place our contributions. WSN related research is immense. We should frankly confess that our concentration is mainly on the ones with a mathematical programming and optimization approach.

2.2. Sensor Placement

In Dhillon and Chakrabarty [8] it is assumed that the sensor field is made up of grid points. It is also assumed that the probability of detection of a target by a sensor decays exponentially with the distance between the target and the sensor. The goal of the proposed sensor placement algorithms is to determine the minimum number of sensors and their locations such that every grid point is covered with a minimum confidence level. The first proposed procedure attempts to maximize the average coverage of the grid points. The second proposed procedure attempts to maximize the coverage of the grid point that is covered least effectively. A theorem proved in the paper provides a sufficient condition under which the nongrid points are adequately covered by the proposed algorithms. The major drawback of the paper is that although the proposed methods provide coverage for the sensor field, they completely ignore energy consumption and network lifetime issues.

In Chakrabarty et al. [9] an integer linear program (ILP) that determines the optimum sensor locations is proposed. The objective is the minimization of the total sensor deployment cost. The constraint is the coverage of the grid points with the required quality. As in Dhillon and Chakrabarty [8] the energy consumption in sensing and processing, in routing the data packets are ignored in the paper. Moreover, an

effective solution method for large instances of the ILP is not provided.

In Yang et al [10] the sensor network is modelled as a graph with the set of vertices being the set of sensors. An edge exists between two nodes if the two corresponding sensors are within the eachother's communication range. An ILP that determines elements of a subset of the vertices is proposed. The objective is the minimization of the number of elements in the subset subject to coverage requirement constraints. A solution procedure based on constructing a feasible solution from the linear programming (LP) relaxation solution of the ILP is also proposed. Although the proposed method finds a set of sensors that cover the sensor field with the required quality, it does not explicitly determine the activity schedules of the sensors, sensor-to-sink data flow routes in order to maximize the network lifetime.

In Altinel et al. [11] the problem of determining sensor locations such that the sensor field is covered with the required quality is addressed. The ILP proposed for this problem aims to minimize total deployment cost. In addition to the classical coverage constraint, the placement of at most one sensor at any point of the sensor field is allowed. They, however, disregard this restriction in their succeeding work (Altinel et al. [12]) where they consider minimum cost coverage problem for heterogeneous WSN, with differentiated coverage quality constraints. In both works they explain how the same ILP models can be used to deal with imperfect and probabilistic sensing in addition to perfect sensing. Greedy and Lagrangean heuristics are proposed in order to solve large instances efficiently. The major drawback of these papers is that they ignore the issues of energy consumption and network lifetime. Wang and Zhong [13] also consider the problem described in the first work of Altinel et al. [11]. Although they develop the same ILP formulation their solution approach is different. They construct a feasible solution for the ILP using an optimal solution of its LP relaxation.

Ganesan et al. [14] formulate an optimization problem that considers jointly sensor placement, transmission structure and data structure in a data gathering sensor network, in terms of an energy related cost function. They study in detail the one dimensional case, namely they provide a closed-form solution for the sensor placement

when the data is independent, and outline the methodology to solve numerically the case when aggregation at nodes results in joint entropy coding, for arbitrary correlation structures. They use their insights from the one dimensional setting to propose an approximation algorithm that places sensors in a two dimensional radial wheel structure. They show that significant power gains can be obtained with such a node placement scheme over commonly used uniform random placements. Although they rigorously consider the sensor placement problem, the authors ignore sensor activity scheduling issue.

Osais et al. [15] introduce a new wireless sensor network planning problem referred to as the Optimal Sensor Configuration (OSC) problem. In this problem, the goal is to find an optimal subset of locations where directional sensors and base stations are installed in order to minimize the total network cost while satisfying the coverage and connectivity requirements. This goal is achieved by appropriately choosing the base station type and configuring each sensor to be installed in the sensor field. The optimal configuration of each sensor is determined by three parameters which are sensing range, field of view and orientation. The paper also gives an integer linear programming formulation of the OSC problem. The viability and effectiveness of the proposed formulation are illustrated through numerical results.

2.3. Sensor Activity Scheduling

In Nakamura et al. [16] the problem of determining activity schedules of the sensors is addressed. A MILP with the objective of minimizing the total energy consumption is formulated. Constraints of the MILP ensure coverage of the sensor field with the required quality and connectivity of the network in each time period. The MILP optimally determines which sensor is active in which period. In the experimental study section it is shown that the network lifetime value obtained using the MILP is much higher than that obtained without performing any sensor activity scheduling. One drawback of the proposed model is that it does not explicitly consider the routing energy. Sensor placement issue is also disregarded.

In Cardei et al. [17] the problem of covering a set of targets with a set of sensors is considered. Only a subset of the sensors that can cover all of the targets are active at any given time, others are in sleep mode. A mixed integer nonlinear program (MINLP) is proposed to maximize the coverage lifetime by determining the sensor subsets and their active time. The constraints guarantee that the active time of a sensor is not greater than an upperbound and every target is covered by the active subset. The major drawback of the paper is that it does not explicitly consider the energy consumed in routing the data packets. Moreover, there is no lower bound on the active time of a sensor subset. One other drawback is that the issue of sensor placement is ignored in the paper. Nanez et al. [18] work on the same problem. They develop a distributed algorithm to obtain suboptimal solutions in an online fashion for large-scale settings. They use game theory in developing the algorithm. The same problem is also considered by Alfieri et al. [19]. Their mathematical model takes explicitly into account the energy consumption in data routing. They use a column generation algorithm to find active subsets and their active periods.

In Ha et al. [20] sensor network is modelled as an undirected connected graph. At any given time only one of the connected subgraphs rooted at the sink is active, others are in sleep mode in order to save energy. An ILP that determines the connected subgraphs is proposed. The objective of this ILP is the minimization of the shared sensors. The constraints guarantee that each sensor belongs to at least one subgraph, the subgraph is connected and rooted at the sink. Also for each sensor, the number of neighbors belonging to the same subgraph has a limit so that the sensor is close to the active subgraph when the subgraph it belongs to is in sleep mode. One drawback of the proposed ILP is that it does not consider coverage of the sensor field. Moreover the energy spent on sensing, processing and routing is not taken into account.

In Liu et al. [21] each sensor can watch at most one target at any given time, and each target should be watched by a predetermined number of sensors. The problem is to find a schedule that meets these requirements. The objective is to maximize lifetime of surveillance subject to energy restrictions. A polynomial time algorithm that finds an optimal solution of the described problem is proposed. The major drawback

of the paper is that it ignores the energy consumption in data routing. Moreover the assumption that each sensor can watch at most one target at any given time is restrictive. The same problem is also considered in Zhao and Gurusamy [22]. They take into account the energy consumption in data routing and find sensor-to-sink data flow paths.

Ai and Abouzeid [23] study a novel “coverage by directional sensors” problem with tunable orientation on a set of discrete targets. They propose a maximum coverage with minimum sensors problem in which coverage in terms of the number of targets to be covered is maximized whereas the number of sensors to be activated is minimized. They present its exact ILP formulation and an approximate (but computationally efficient) centralized greedy heuristic. These centralized solutions are used as baselines for comparison. Then they provide a distributed greedy algorithm (DGA) solution. By incorporating a measure of the sensors residual energy into DGA, they further develop a Sensing Neighborhood Cooperative Sleeping (SNCS) protocol which performs adaptive scheduling on a larger time scale. Finally, they evaluate the properties of the proposed solutions and protocols in terms of providing coverage and maximizing network lifetime through extensive simulations. Moreover, for the case of circular coverage, they compare against the best known existing coverage algorithm. They disregard sensor and sink placement and data routing issues.

Yan et al. [24] introduce an adapting sensing coverage mechanism for differentiated surveillance in sensor networks. Unlike previous schemes, this one guarantees not only full sensing coverage to a certain geographic area, but also the degree of coverage up to the limit imposed by the density of available nodes. In the protocol, each node is able to dynamically decide a schedule for itself to guarantee a certain degree of coverage (DOC) with average energy consumption inversely proportional to the node density. The distinguishing advantages of their scheme are a much longer network half-life through energy conservation and balancing among sensor nodes, and a small communication overhead required to establish a working duty schedule among nodes. Several optimizations and extensions are proposed to provide even better performance. Simulations show that their protocol accomplishes differentiated surveillance with low

energy consumption and outperforms other state-of-the-art schemes by as much as a 50% reduction in energy consumption and as much as a 130% increase in the half-life of the network. Unfortunately they do not consider sensor and sink placement, and data routing issues.

Yardibi and Karasan [25] develop the distributed adaptive sleep scheduling algorithm (DASSA) for WSNs with partial coverage. DASSA does not require location information of sensors while maintaining connectivity and satisfying a user defined coverage target. In DASSA, nodes use the residual energy levels and feedback from the sink for scheduling the activity of their neighbors. This feedback mechanism reduces the randomness in scheduling that would otherwise occur due to the absence of location information. The performance of DASSA is compared with an integer linear programming based centralized sleep scheduling algorithm (CSSA), which is devised to find the maximum number of rounds the network can survive assuming that the location information of all sensors is available. DASSA attains network lifetimes up to 92% of the centralized solution.

2.4. Data Routing

Younis et al. [26] assume that the sensors and the sink are connected by bidirectional wireless links with a cost associated with each direction. Each link may have a different cost for each direction due to different energy levels of the nodes at each end. The cost function for a link between two sensors has components that take into account the distance between them and the remaining energy of the receiver sensor. The cost of a path between two sensors is defined as the sum of the costs of the links traversed. For each sensor that collects data the proposed routing algorithm finds a least-cost path from the sensor to the sink using Dijkstra's algorithm. The major drawback of the proposed routing algorithm is that it considers each sensor separately. It does not find a global minimum for the energy consumption of all the sensors while routing the data packets.

In Sankar and Liu [27] a LP formulation is presented. The objective is the max-

imization of the network lifetime. The first constraint is the flow balance equation. The number of packets received by a sensor should be equal to the number of packets transmitted to other sensors. The second constraint is the energy limitation. The energy spent in transmitting the data packets should be less than the battery energy. A distributed routing algorithm that reaches an optimal solution to within an asymptotically small relative error is proposed. The major drawback of this work is that it ignores sensor placement and sensor activity scheduling issues. Also energy spent in receiving the data packets is not taken into account.

In Chang and Tassiulas [28] the routing problem is formulated as a LP problem. The objective is the network lifetime maximization. The constraints are flow balance and battery energy limitation. Two different models are considered for the information-generation processes. One assumes constant rates and the other assumes an arbitrary process. A shortest path routing algorithm is proposed which uses link costs that reflect both the communication energy consumption rates and the residual energy levels at the two end nodes. The algorithm is amenable to distributed implementation. Simulation results with the information-generation process models show that the proposed algorithm can achieve network lifetime that is very close to the optimal network lifetime obtained by solving the LP problem.

Xue et al. [29] study energy efficient routing for data aggregation in wireless sensor networks. The goal is to maximize the lifetime of the network given the energy constraint on each sensor node. This problem is modelled as a linear multicommodity flow problem where a commodity represents the data generated from a sensor node and delivered to a sink. A fast approximate algorithm is presented. Then along this baseline several advanced topics are studied. First an algorithm, which utilizes the unique characteristic of data aggregation is designed. It is proved that it reduces the running time of the fastest existing algorithm by a factor of the number of commodities. Second, the proposed algorithm is extended to accommodate the same problem in the setting of multiple base stations and its impact on network lifetime improvement is studied.

In Hua and Yum [30] the problem of routing the data to the sink node in a way to maximize network lifetime is considered. Network lifetime is defined as the shortest lifetime of all the nodes. A node can only use its downstream neighbors (downstream with respect to the sink) to route its traffic. The flow coming from an upstream node j is reduced at node i because some of the information coming from node j is the same as those coming from other neighbors of i . The problem is first formulated as a mathematical program where the objective is the minimization of maximum normalized nodal power consumption for the bottleneck nodes. Normalization is done by nodal battery energy. The constraints are linear flow conservation constraints. This min-max objective function is not differentiable. Hence a smoothing function is used. The smoothed objective function is approximately equivalent to the original objective function. The Karush-Kuhn-Tucker conditions are not sufficient for optimality for this reason a set of sufficient conditions and a gradient descent algorithm for a distributed application are developed. At each iteration of the algorithm each node adjusts its downstream flow in the direction of the gradient until each flow link satisfies the sufficient optimality conditions. According to the simulation results it is possible to say that the algorithm converges efficiently.

Ergen and Varaiya [31] present a routing algorithm that maximizes the lifetime of a sensor network by adjusting the number of packets traversing each node. The adjustment is carried out by transmitting over alternative routes. The first part of the paper assumes that the worst delay resulting from energy efficient routing is less than the maximum tolerable value. Ignoring the delay constraint of the network the routes are selected as the solution to a LP problem in which the objective is to maximize the minimum lifetime of each node. The solution is implemented in a centralized algorithm and then approximated by an iterative algorithm based on least cost path routing in which each step is implemented efficiently in a distributed manner. The second part of the paper incorporates the delay guarantee into energy efficient routing by constraining the length of the routing paths from each sensor node to the collection node. Simulations reveal that the lifetime of the network increases significantly by optimal routing and including delay constraint in energy efficient routing improves the network performance since the delay of the network keeps increasing as the delay

constraint is relaxed beyond the value at which the optimal lifetime is achieved.

In Pham et al. [32] sensor network is divided into a grid structure. It is assumed that each grid cell can be covered by any sensor in the cell. It is also assumed that a sensor in a cell can transmit data packets to any sensor in the adjacent cell. After these simplifying assumptions the problem is to find the total data flow among the cells to maximize the lifetime until the first cell loses area coverage. A LP with the objective of network lifetime maximization is proposed. The constraints are flow balance and battery energy limitation. These constraints are written for the cell not for each sensor. Simulations show that the proposed method increases network lifetime.

In Ciciriello et al. [33] the problem of routing data efficiently from multiple sources to multiple sinks is presented. An ILP which minimizes the number of links used in transmitting data from the sources to the sinks is proposed. The constraints guarantee a connected, end-to-end path for each source-sink pair. A novel decentralized scheme that adapts the topology by maximizing the overlapping among source-sink paths, therefore minimizing the overall number of network links exploited is proposed. The major drawback of the paper is that it does not explicitly consider the energy consumption in sensing and processing the data and in routing the data packets.

Cui et al. [34] studies the problem of distributed maximum network lifetime routing for a multihop wireless network. Inspired by the max-min fair resource allocation, the paper presents a novel utility-based nonlinear formulation of this problem. Based on this formulation, this paper further presents a distributed routing algorithm that achieves the goal of maximizing network lifetime. The presented algorithm has a solid nonlinear optimization theoretical background and is shown to be effective and efficient under various simulation environments.

In Norman et al. [35] a faster energy efficient load balancing protocol for routing is proposed. It does fairly well compared to popular protocols Dynamic Source Routing (DSR) and Ad Hoc on Demand Distance Vector (AODV) when simulated in GloMoSim environment. The proposed model has less time delay, more energy and

better distributed work mode. This can be very effective for detecting environmental changes where sensors have a fixed location. The limitation in this protocol is that it does not guarantee the delivery of all the packets generated as retransmission is not supported. So there is a very small probability of missing relevant data.

2.5. Sink Location

In Bogdanov et al. [7] the problem of positioning data collecting base stations in a sensor network is considered. In the paper it is shown that the choice of positions has a marked influence on the data rate, or equivalently on the power efficiency of the network. The optimum data rate for a fixed layout of base stations can be found by a maximum flow algorithm. Finding the optimum layout of base stations, however, turns out to be an NP-complete problem, even in the special case of homogeneous networks. The analysis of the optimum layout for the special case of the regular grid shows that all layouts that meet certain constraints are equally good. Two classes of random graphs chosen to model networks that might be realistically encountered are also considered. The performance of several base station positioning algorithms on instances of these classes are evaluated. In the paper it is shown that in comparison to manually choosing positions along the periphery of the network or randomly choosing them within the network, the algorithms tested find positions which significantly improve the data rate and power efficiency of the network.

Vincze et al. [36] give a mathematical model that determines the locations of the sinks minimizing the sensors' average distance from the nearest sink. An iterative algorithm that is able to find the sink locations given by the mathematical model is presented. However, the algorithm uses global information about the network that is impractical in wide area sensor networks. Thus a novel iterative algorithm that carries out the sink deployment based only on the location information of the neighboring nodes is proposed. These two algorithms are compared and it is shown that the performance of the second one is very close to the performance of the first one. It is also argued that the neighboring nodes of the sinks have a high traffic load, thus the lifetime of the network can be elongated by relocating the sinks from time to time.

Consequently a relocation algorithm for the coordinated relocation of multiple sinks is proposed. The simulation results show that the algorithm extend the network lifetime significantly. One drawback of the paper is that although sinks are optimally located, the energy spent in sensing and processing and in routing the data packets are not explicitly considered.

In Basagni et al. [37] a mobile sink is considered. The problem is to determine the starting site and the route for the mobile sink together with the sojourn times of the sink at each visited site so that the network lifetime is maximized. First a MILP formulation is proposed. The objective is the maximization of the sink's total time at sojourning sites. The first constraint guarantees that the total energy spent in receiving and transmitting packets and in setting up/releasing routes when the sink moves to a site for each sensor is less than the battery energy. The remaining constraints ensure the formation of a path for the sink and eliminate cycles that can be obtained. In the second part of the paper the first heuristics for controlled sink movements that are fully distributed and localized are described. Simulations show that the heuristics increase the network lifetime considerably. One drawback of the paper is that it does not explicitly include the energy consumption in sensing and processing and in routing the data packets in the MILP formulation. Besides the sink can do only one hour during the network lifetime, which is fairly unrealistic.

2.6. Works That Integrate Some of the Design Issues

The second MILP model proposed by Patel et al. [38] integrates placement, data routing and sink location issues. The objective is the minimization of the total cost of placing the sensors and the sinks. The constraints guarantee the coverage of the sensor field with the required quality and flow conservation. The MILP finds optimally sensor and sink locations and sensor-to-sink data flow quantities. They also develop another MILP model with network lifetime maximization objective. In this MILP in addition to the coverage and flow balance constraints there is the energy constraint that limits the energy spent in data routing.

In Hou et al. [39] placement and data routing issues are integrated. The problem is to find the locations of a given number of relay sensors, to allocate a given amount of energy to them and to determine sensor-to-sink flow quantities. A mixed-integer nonlinear program (MINLP) with the objective of network lifetime maximization is proposed. The constraints ensure flow balance and the energy limitation in receiving and transmitting the data packets. Solving the MINLP is computationally difficult. Therefore, a heuristic algorithm is developed. Through numerical results it is shown that the heuristic algorithm offers a very attractive solution and some important insights to the problem addressed.

Gatzianas and Georgiadis [40] integrate data routing and sink location. Mobile sink case is considered. The problem is to find the sink sojourn times and the routing flow vector for each sink location that maximize the network lifetime. A mathematical programming formulation is presented. The objective is the network lifetime maximization defined as the sum of sink sojourn times. The constraints guarantee flow balance and ensure that the battery energy is not exceeded in routing the data packets. A distributed algorithm based on the subgradient method and using the sink as leader is proposed. Efrat et al. [41] consider the same problem. They compute a location for the sink and a transmission scheme that approximates up to a constant factor the lifespan of the optimal solution by solving n linear programming instances. Shi and Hou [42] also study the same problem. The computational complexity of the algorithm they propose is smaller than that of proposed by Efrat et al. [41]. Shi et al. [43] extend the problem to address the network capacity maximization in a later work.

Kim et al. [44] integrate data routing and sink location. They consider a static sink. They formulate a MILP where the objective maximizes the minimum amount of data produced by each sensor and the total data packets generated. The MILP optimally determines the locations of a predetermined number of sinks, the sensor-to-sink flow routes and the data volume produced by each sensor. The constraints are flow balance and battery energy limitation restrictions. Unfortunately an efficient solution procedure for the MILP is not presented.

In this thesis we first integrate sensor placement and sensor activity scheduling problems. To the best of our knowledge it is the first time in the literature that this integration is attempted. We develop a MILP model with the objective of lifetime maximization. The constraints guarantee the coverage of the sensor field with the required quality levels, ensure that the energy spent over all time periods is less than the battery energy for each sensor and total cost of the placed sensors is less than the budget level. The MILP optimally determines which sensor is active in which period and the optimal sensor locations. Next, we integrate all of the four issues namely, sensor placement, sensor activity scheduling, data routing and sink location with an MILP formulation. Again to the best of our knowledge it is the first time in the literature that this integration is realized.

3. LOCATION AND SCHEDULING PROBLEM

3.1. Introduction

In this chapter¹ we present a MILP model and a solution procedure for the *Location and Scheduling Problem* (LSP). LSP consists of the determination of the location of the sensors and their activity schedules in order to maximize the network lifetime subject to coverage requirement, energy restrictions and budget constraint.

3.2. Model Formulation

We consider a sensor field consisting of points indexed by $j = 1, \dots, J$ that should be covered by sensors after their deployment over these J points. In the sequel we use both i and j as indices for sensor locations. We assume that there are K different types of sensors. A sensor of type k has sensing range r_k^s , energy consumption rate e_k in active mode, initial battery energy E_k , and unit cost c_k . The total available budget for sensors is limited by B monetary units. Since the objective of the model is the maximization of the coverage lifetime L , we assume a sufficiently long planning horizon which consists of T periods of equal length. In other words, T becomes an upper bound for L . In each period $t = 1, \dots, T$, a sensor is either in active or standby mode, and energy expenditure of a sensor in standby mode is negligible (xbow [46]). It is also realistic to assume that the energy saved while a sensor is in standby mode is higher than the energy required for activating the sensor.

We develop a mathematical programming formulation to determine the points on which sensors have to be placed, and the active/inactive periods of the sensors such that the coverage lifetime L of the sensor network is maximized. Since it includes the set covering problem as a special case, which is known to be NP-complete (Garey and Johnson [47]), LSP is expected to be even more difficult to solve exactly.

¹The paper Türkoğulları et al. [45] based on this chapter is published in the proceedings of ISCIS 2007 and received the best paper award in computer networks and systems track.

First, we define the decision variables. x_{ik} is equal to one if a type- k sensor is placed at point i , it is equal to zero otherwise. z_{ikt} is equal to one if a type- k sensor placed at point i is active in period t , it is equal to zero otherwise. y_t is equal to one if period t is within lifetime L , it is equal to zero otherwise. L is a nonnegative integer variable that denotes the network lifetime.

We let a_{ijk} denote the parameter whose value is determined according to sensing characteristics. For perfect sensing, $a_{ijk} = 1$ if a sensor of type k placed on point i can cover point j , otherwise $a_{ijk} = 0$. For imperfect and uncertain sensing, it may take on fractional values in the interval $[0, 1]$ as explained in Altinel et al [12]. d_j is the given coverage quality requirement. Its value also depends on the sensing characteristics and the type of the coverage. It is a positive integer for perfect sensing and becomes nonnegative rational number for imperfect and probabilistic sensing. The new model is given as:

$$\text{Max } L \quad (3.1)$$

s.t.

$$0 \leq L \leq T \quad (3.2)$$

$$\sum_{i=1}^J \sum_{k=1}^K a_{ijk} z_{ikt} \geq d_j y_t \quad j = 1, \dots, J; t = 1, \dots, T \quad (3.3)$$

$$T y_t \geq L + 1 - t \quad t = 1, \dots, T \quad (3.4)$$

$$\sum_{t=1}^T z_{ikt} \leq \left\lfloor \frac{E_k}{e_k} \right\rfloor x_{ik} \quad i = 1, \dots, J; k = 1, \dots, K \quad (3.5)$$

$$z_{ikt} \leq x_{ik} \quad i = 1, \dots, J; k = 1, \dots, K; t = 1, \dots, T \quad (3.6)$$

$$\sum_{k=1}^K c_k \sum_{i=1}^J x_{ik} \leq B \quad (3.7)$$

The objective function (3.1) maximizes the coverage lifetime L . Constraint (3.2) says that planning horizon T is an upper bound for L . Constraints (3.3) and (3.4) ensure that in period t within the lifetime, the coverage level must be at least equal to d_j for all points j . Note that when $y_t = 1$, coverage constraint is active, whereas it is redundant when $y_t = 0$. Constraint (3.5) ensures that the total number of active periods of

a type k sensor placed at point i cannot exceed the maximum possible number of periods that is obtained by the ratio of battery energy to the energy consumption rate. Constraints (3.6) guarantee that a sensor cannot be active without being located first. Constraint (3.7) makes sure that the total cost of the deployed sensors remains within the budget. Observe that this model consists of $T(J+1) + KI(T+1) + 2$ constraints, $KI(T+1) + T$ binary variables and one nonnegative continuous variable.

3.3. Solution Procedure

The developed mixed-integer linear programming model is difficult to solve for large problem instances. Therefore, we propose a heuristic solution procedure which is based on Lagrangean relaxation of the formulation (3.1)–(3.7). When the coverage constraints (3.3) and the budget constraint (3.7) are relaxed, i.e., removed from the constraint set and added to the objective function after being multiplied by nonnegative multipliers $\boldsymbol{\mu}$ and λ respectively, we obtain the following Lagrangean problem:

$$\begin{aligned}
 L_{UB}(\boldsymbol{\mu}, \lambda) = \max L + \lambda \left(B - \sum_{i=1}^J \sum_{k=1}^K c_k x_{ik} \right) \\
 + \sum_{t=1}^T \sum_{j=1}^J \mu_{jt} \left[\left(\sum_{i=1}^J \sum_{k=1}^K a_{ijk} z_{ikt} \right) - d_j y_t \right] \\
 \text{s.t.} \quad (3.2), (3.4)–(3.6).
 \end{aligned} \tag{3.8}$$

By defining $\beta_{ikt} = \sum_{j=1}^J \mu_{jt} a_{ijk}$, $\gamma_{ik} = \lambda c_k$, $\alpha_t = \sum_{j=1}^J \mu_{jt} d_j$, and $\Phi = \lambda B$, and manipulating the summation terms, we obtain the following equivalent expression:

$$\begin{aligned}
 L_{UB}(\boldsymbol{\mu}, \lambda) = \max L + \sum_{i=1}^J \sum_{k=1}^K \sum_{t=1}^T \beta_{ikt} z_{ikt} \\
 - \sum_{i=1}^J \sum_{k=1}^K \gamma_{ik} x_{ik} - \sum_{t=1}^T \alpha_t y_t + \Phi.
 \end{aligned} \tag{3.9}$$

It can be shown that $L_{UB}(\boldsymbol{\mu}, \lambda) \geq L$ for all values of $\boldsymbol{\mu} \geq \mathbf{0}$ and $\lambda \geq 0$. Hence, our

aim is to find the smallest possible value of $L_{UB}(\boldsymbol{\mu}, \lambda)$ denoted as L_{UB}^* . This problem is called Lagrangean dual problem and is given as

$$L_{UB}^* = \min_{\boldsymbol{\mu} \geq \mathbf{0}, \lambda \geq 0} L_{UB}(\boldsymbol{\mu}, \lambda) \quad (3.10)$$

It is easy to see that for given values of $\boldsymbol{\mu} \geq \mathbf{0}$ and $\lambda \geq 0$ the Lagrangean problem can be separated into two subproblems that can be solved independently. The first subproblem is given as

$$\begin{aligned} L_{UB1}(\boldsymbol{\mu}, \lambda) = \max & \sum_{i=1}^J \sum_{k=1}^K \sum_{t=1}^T \beta_{ikt} z_{ikt} - \sum_{i=1}^J \sum_{k=1}^K \gamma_{ik} x_{ik} \\ \text{s.t.} & \quad (3.5), (3.6). \end{aligned} \quad (3.11)$$

Note that it is separable for each k and i . In other words, we can determine whether or not a type k sensor has to be placed on point i to maximize $L_{UB1}(\boldsymbol{\mu}, \lambda)$. If there is no sensor of type k on point i ($x_{ik} = 0$), then $z_{ikt} = 0$ for all t by (3.5) and (3.6), which makes the objective value equal to zero. Otherwise ($x_{ik} = 1$), the maximum number of periods in which the sensor is active is given by $\lfloor E_k/e_k \rfloor$. In this case, the selection of z_{ikt} 's whose value must be equal to one should be determined based on the coefficients β_{ikt} in such a way that the objective function is maximized. This is done by first sorting the coefficients in nondecreasing order $\beta_{ikt[1]} \geq \beta_{ikt[2]} \geq \dots \geq \beta_{ikt[T]}$, and then setting $z_{ikt[l]} = 1$ for $l = 1, \dots, \lfloor E_k/e_k \rfloor$. If $\sum_{l=1}^{\lfloor E_k/e_k \rfloor} \beta_{ikt[l]} - \gamma_{ik} > 0$, then an optimal solution is given by $x_{ik} = 1$, $z_{ikt[l]} = 1$ for $l = 1, \dots, \lfloor E_k/e_k \rfloor$, and $z_{ikt[l]} = 0$ for $l = \lfloor E_k/e_k \rfloor + 1, \dots, T$. If $\sum_{l=1}^{\lfloor E_k/e_k \rfloor} \beta_{ikt[l]} - \gamma_{ik} \leq 0$, an optimal solution is obtained by setting $x_{ik} = 0$ and $z_{ikt[l]} = 0$ for $l = 1, \dots, T$ resulting in zero objective value. Figure 3.1 summarizes this procedure that solves the first subproblem.

-
1. For each sensor of type k located on point i Do
 - 1.1 Sort β_{ikt} in nondecreasing order as

$$\{\beta_{ikt[1]} \geq \beta_{ikt[2]} \geq \dots \geq \beta_{ikt[T]}\}$$
 - 1.2. If $\sum_{l=1}^{\lfloor E_k/e_k \rfloor} \beta_{ikt[l]} - \gamma_{ik} > 0$ Then
 - 1.2.1 Set $x_{ik} = 1$, $z_{ikt[l]} = 1$ for $l = 1, \dots, \lfloor \frac{E_k}{e_k} \rfloor$,
 and $z_{ikt[l]} = 0$ for $l = \lfloor E_k/e_k \rfloor + 1, \dots, T$
 Else $(\sum_{l=1}^{\lfloor E_k/e_k \rfloor} \beta_{ikt[l]} \leq \gamma_{ik})$
 - 1.2.2 $x_{ik} = 0$ and $z_{ikt[l]} = 0$ for $l = 1, \dots, T$
 - End If
 - End For
-

Figure 3.1. Solution algorithm for the first subproblem

The second subproblem is given as

$$\begin{aligned}
 L_{UB2}(\boldsymbol{\mu}, \lambda) &= \max L - \sum_{t=1}^T \alpha_t y_t \\
 \text{s.t. } & (3.2), (3.4).
 \end{aligned} \tag{3.12}$$

whose solution can also be found by inspection. Notice that if the value of L is known, which is in the interval $[0, T]$, then optimal values of y_t can easily be determined. Suppose that $L = t_0 - 1$. Then $y_t = 1$ must hold for $1 \leq t < t_0$ due to constraint (3.4). Furthermore, since $\alpha_t \geq 0$, $y_t = 0$ for $t_0 \leq t \leq T$ to yield the maximum possible objective value. As a result, we can try all possible values for L , and select the one providing the highest objective value. The procedure is outlined in Figure 3.2.

Once these subproblems are solved, $L_{UB}(\boldsymbol{\mu}, \lambda)$ can be computed as $L_{UB}(\boldsymbol{\mu}, \lambda) = L_{UB1}(\boldsymbol{\mu}, \lambda) + L_{UB2}(\boldsymbol{\mu}, \lambda) + \Phi$. In order to solve the Lagrangean dual problem (3.10) and obtain the best (smallest) upper bound L_{UB}^* , we can apply the subgradient algorithm. At each iteration of the subgradient algorithm, the Lagrangean subproblems are first

$$1. \ t_0 = \arg \max_{t \in \{1, \dots, T+1\}} \left\{ t - 1 - \sum_{l=1}^{t-1} \alpha_l \right\}.$$

An optimal solution is given by:

$$y_t = \begin{cases} 1 & 1 \leq t < t_0 \\ 0 & t_0 \leq t \leq T \end{cases} \quad \text{and } L = t_0 - 1.$$

Figure 3.2. Solution algorithm for the second subproblem

solved to obtain $L_{UB}(\boldsymbol{\mu}, \lambda)$ with the current values of the multipliers $\boldsymbol{\mu}$ and λ . Their values are then updated and the Lagrangean subproblems are solved again.

A crucial issue in the subgradient algorithm is the update of the multipliers. It requires generating a feasible solution for the original problem, which can be done by utilizing the solution of the Lagrangean problem. This is achieved by applying at each iteration of the subgradient algorithm a heuristic to transform the solution of the Lagrangean problem that is possibly infeasible with respect to the relaxed constraints into a feasible solution of the original problem. This means that a lower bound is generated for the original problem denoted by L_{LB} . The best (greatest) lower bound L_{LB}^* attained at the end of the iterations and the corresponding solution determine the performance of the heuristic. As a result, Lagrangean heuristic provides us with both a lower and an upper bound for the original problem. The optimal objective value L^* should be contained in the interval $[L_{LB}^*, L_{UB}^*]$. Clearly, $L_{LB}^* = L_{UB}^*$ implies that we have found the optimal value. When $L_{LB}^* < L_{UB}^*$, we can evaluate the quality of the feasible solution at hand, which is an important characteristic of Lagrangean heuristics.

There are two ways to terminate the iterations of the subgradient algorithm. The first one is based on the gap between L_{LB}^* and L_{UB}^* . Iterations are stopped if $L_{UB}^* - L_{LB}^* \leq \varepsilon_1$ where ε_1 is a predetermined small positive number. Another way is to base the termination criterion on parameter θ , which keeps track of the number of iterations without improving the best upper bound L_{UB}^* . The value of θ is set to two initially and it is halved whenever there is no improvement in L_{UB}^* for consecutive

N iterations. When $\theta < \varepsilon_2$, iterations are stopped. In our application, we use both of these criteria with $\varepsilon_1 = 0$, $\varepsilon_2 = 0.001$, and $N = 30$. That is, iterations are either terminated when $L_{UB}^* = L_{LB}^*$ (optimal solution is found), or $\theta < 0.001$. We summarize the Lagrangean heuristic in Figure 3.3.

Finally, we elaborate on the heuristic used for generating the feasible solution at each iteration. The main idea is to start with the solution of the Lagrangean problem and make adjustments in the values of the decision variables so that the solution satisfies the relaxed constraints (3.3) and (3.7). For each period $t = 1, \dots, L_{UB}^*$, we carry out the following procedure. If coverage is not satisfied in period t , the uncovered level u_{jt} of each point j in period t is calculated first. It is defined as

$$u_{jt} = \max \left(0, d_j - \sum_{i=1}^J \sum_{k=1}^K a_{ijk} z_{ikt} \right). \quad (3.13)$$

For each potential point i in the grid, we check the effect of two strategies: (i) Activating a sensor on point i that is inactive in period t , and (ii) placing a new sensor on point i , and activate it. It is reasonable to give priority to the first strategy since the second one reduces the remaining budget by deploying a new sensor. Consequently, the one having the greatest coverage-energy product p among the already deployed sensors is selected for activation. p is defined as

$$p_{ikt} = \sum_{j=1}^J a_{ijk} u_{jt} g_{ikt}, \quad (3.14)$$

where g_{ikt} is the remaining energy (in terms of periods) of a type k sensor placed on point i in period t . The activation of the standby sensors is continued until either all existing sensors run out of energy or there is no reduction in the uncovered level due to the activation of existing sensors. Then, the second strategy is implemented. Namely, the best point is determined to deploy a new sensor. This is done by computing the largest coverage-cost ratio r defined with (3.15) for each sensor type k and grid point

i. The one having the largest ratio is selected for deployment and activation.

$$r_{ikt} = \frac{\sum_{j=1}^J a_{ijk} u_{jt}}{c_k} g_{ikt}. \quad (3.15)$$

After determining the sensor in the standby mode to be activated or to be newly

1. Initialization: Set the iteration counter $s = 0$

$$\mu_{jt}^{(s)} = 1, \forall j, t; \lambda^{(s)} = 1; \theta^{(s)} = 2$$

2. While $|L_{UB}^* - L_{LB}^*| > \varepsilon_1$ and $\theta > \varepsilon_2$ Do

2.1. Solve subproblems (3.11), (3.12), and compute $L_{UB}^{(s)}$

2.2. Update L_{UB}^* as $L_{UB}^* = \min \{L_{UB}^*, L_{UB}^{(s)}\}$

2.3. If L_{UB}^* is not updated during N iterations Then
set $\theta \leftarrow \theta/2$, and switch to the stored μ, λ values

End If

2.4. Construct a feasible solution with objective value $L_{LB}^{(s)}$
using the algorithm in Figure 3.4

2.5. Update L_{LB}^* as $L_{LB}^* = \max \{L_{LB}^*, L_{LB}^{(s)}\}$

2.6. Update μ, λ as

$$\begin{aligned} \mu_{jt}^{(s+1)} &= \max \left\{ \mu_{jt}^{(s)} - \eta^{(s)} \left[\left(\sum_{i=1}^J \sum_{k=1}^K a_{ijk} z_{ikt} \right) - d_j y_t \right], 0 \right\} \\ \lambda^{(s+1)} &= \max \left[\lambda^{(s)} - \eta^{(s)} \left(B - \sum_{i=1}^J \sum_{k=1}^K c_k x_{ik} \right), 0 \right] \end{aligned}$$

where $\eta^{(s)} = \frac{\theta^{(s)} (L_{UB}^{(s)} - L_{LB}^*)}{\sum_{t=1}^T \sum_{j=1}^J \left\{ \left(\sum_{i=1}^J \sum_{k=1}^K a_{ijk} z_{ikt} \right) - d_j y_t \right\}^2 + \left(B - \sum_{i=1}^J \sum_{k=1}^K c_k x_{ik} \right)^2}$

2.7. $s \leftarrow s + 1$

End While

Figure 3.3. Lagrangean Heuristic

deployed, the uncovered levels are updated and the procedure is repeated until the coverage constraint is satisfied for period t . If the budget is not sufficient to locate a new sensor, we remove some other sensors that are not used until period t . We give

the steps of the overall procedure in Figure 3.4.

1. Close all sensors redundant for coverage
 2. $t = 1$
 3. While $t \leq T$ Do
 - 3.1. If every point is covered Then
 - 3.1.1 If the budget constraint is satisfied Then
 - $L \geq t, t \leftarrow t + 1$
 - Else
 - 3.1.2. Remove unused sensors in $[0, t]$ starting with the largest cost until the budget constraint is satisfied. If this is not possible then set $L = t - 1$ and stop
 - End If
 - 3.2. Else (every point is not covered)
 - 3.2.1. Find the sensor with the highest p value (p^*)
 - 3.2.2. If $p^* > 0$ Then activate that sensor
 - 3.2.3. Else find the sensor with the highest r value (r^*)
 - 3.2.4. If $r^* = 0$ Then $L = t - 1$ and stop
 - Else
 - 3.2.4.1. If budget is enough Then activate the sensor
 - 3.2.4.2. Else remove unused sensors in $[0, t]$ starting with the largest cost until the budget constraint is satisfied. If this is not possible then set $L = t - 1$ and stop
 - End If
 - End If
 - End If
 - End While
-

Figure 3.4. Generation of a feasible solution

4. A NEW VALID INEQUALITY AND CUTTING PLANE FOR COVERING BINARY INTEGER PROGRAMS

4.1. Introduction

A covering binary integer programming problem (IPC) can be formulated as

$$\text{IPC} : \min \{ \mathbf{c}^T \mathbf{x} : \mathbf{A}\mathbf{x} \geq \mathbf{b}, \mathbf{x} \in \{0, 1\}^n \}.$$

Here, $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{c}$ and $\mathbf{b}$ are respectively n and m dimensional vectors. We assume that the entries of $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{c}$ are nonnegative real numbers.

Consider the inequalities

$$\mathbf{A}\mathbf{x} \geq \mathbf{b} \tag{4.1}$$

$$\mathbf{x} \leq \mathbf{1} \tag{4.2}$$

$$\mathbf{x} \geq \mathbf{0} \tag{4.3}$$

and define the polytope $P(\mathbf{A})$ as

$$P(\mathbf{A}) = \{ \mathbf{x} \in \mathbb{R}^n : \mathbf{x} \text{ satisfies (4.1) - (4.3) } \}. \tag{4.4}$$

Then, the set $S(\mathbf{A}) = P(\mathbf{A}) \cap \mathbb{Z}^n$ denotes the binary vectors in $P(\mathbf{A})$. Define also $C(\mathbf{A})$ as the convex hull of $S(\mathbf{A})$. Notice that $C(\mathbf{A}) \subseteq P(\mathbf{A})$. In (4.2) $\mathbf{1}$ is the n dimensional vector with components equal to 1. In (4.3) $\mathbf{0}$ is the n dimensional vector with components equal to 0. This polytope arises from the coverage constraints (3.3) of the previous chapter. When a_{ij} is 0 or 1 and when $\mathbf{b} = \mathbf{1}$ $C(\mathbf{A})$ becomes the well known set covering polytope ([48]-[57]). The fact that a_{ij} is 0 or 1 and $\mathbf{b} = \mathbf{1}$ in the set covering polytope is a crucial assumption of all these works in the definition and analysis of valid inequalities and cutting planes. For this reason the results presented

for the set covering polytope are not directly applicable to $C(\mathbf{A})$. Nevertheless, there are ideas, proving techniques in the work related to set covering polytope that can be adopted for the analysis of $C(\mathbf{A})$.

In this chapter we first propose a valid inequality for $C(\mathbf{A})$ in a way similar to [50]-[52]. We then present a condition for facet definition. The validity and facet definition conditions may not be fulfilled for $C(\mathbf{A})$ but may be fulfilled for $C(\mathbf{A}_M^{N_r})$. $\mathbf{A}_M^{N_r}$ is the submatrix of $\mathbf{A}$ formed by the row set $M = \{1, 2, \dots, m\}$ of $\mathbf{A}$ and a subset N_r of columns $N = \{1, 2, \dots, n\}$ of $\mathbf{A}$. For this case we present a lifting procedure and prove that the lifted inequality is still valid and bears the facet defining property. We then propose a polynomial time algorithm that produces valid inequalities for $C(\mathbf{A})$ by making use of the proved propositions.

First we consider the set $S(\mathbf{A})$ and obtain a cutting plane for $S(\mathbf{A})$ by using arguments originally proposed for the set covering polytope [48, 49]. We use a variant of Farkas' Lemma to obtain a disjunction which is satisfied by all elements of $S(\mathbf{A})$ giving an objective function value for IPC that is smaller than a given upper bound z^u . The disjunction is valid under certain conditions. Then we propose a polynomial time algorithm that produces a feasible solution $\mathbf{x}^u$ for IPC and a cutting plane separating all points of $S(\mathbf{A})$ having an objective value smaller than $\mathbf{c}^T \mathbf{x}^u$.

Throughout the chapter we denote by M the row set $\{1, 2, \dots, m\}$ of matrix $\mathbf{A}$ in (4.1) and by N its column set $\{1, 2, \dots, n\}$. $\mathbf{A}_{M_1}^{N_1}$ is the submatrix of $\mathbf{A}$ having $M_1 \subseteq M$ as its row set and $N_1 \subseteq N$ as its column set. $\mathbf{I}^n$ is the $n \times n$ identity matrix. z^u is an upper bound value for IPC . $\mathbf{x}^u$ is the feasible solution providing the objective value $z^u = \mathbf{c}^T \mathbf{x}^u$. Matrices are denoted by upper case letters, their elements are denoted by lower case letters. For instance $\mathbf{A}$ is a matrix, and a_{ij} is its element in row i at column j . The i^{th} component of a vector $\mathbf{v}$ is v_i . LP relaxation LPC of problem IPC

is formulated as

$$\min \quad \mathbf{c}^T \mathbf{x} \quad (4.5)$$

$$\text{s.t.} \quad (4.1), (4.2), (4.3). \quad (4.6)$$

We let $\mathbf{x}^{lpc}$ denote one of its optimum solutions and associate the hypergraph $H_A(N, E)$ [59] having N and E as the sets of vertices and edges respectively, with each nonempty integer set $S(\mathbf{A})$. E consists of the subsets e of N satisfying relations

$$\sum_{j \in e} a_{ij} \geq b_i \quad i \in M. \quad (4.7)$$

In other words e represents a subset of columns for which the rowwise addition of its entries (i.e. $\sum_{j \in e} a_{ij}$) is larger than b_i for all row $i \in M$. A similar notation has been used also by Saxena [56] for the set covering problem.

4.2. A Valid Inequality for the Binary Covering Polytope

In this section we propose a valid inequality for $C(\mathbf{A})$, which we assume nonempty. The following proposition provides a sufficient condition for the validity of an inequality.

Proposition 1. *Let N_1, N_2 be two subsets of N , M_1, M_2 be two subsets of M , d be a positive integer. It is assumed that N_1, N_2, M_1, M_2, d satisfy the following conditions: $|N_1| \geq d + 1$, $N_1 \cup N_2 = N$, $N_1 \cap N_2 = \emptyset$, $M_1 \cup M_2 = M$, $M_1 \cap M_2 = \emptyset$. It is also assumed that $a_{ij} = 0$ $i \in M_1, j \in N_2$ if $|N_2| > 0$. If these assumptions are satisfied and if for any $S \subseteq N_1$ with $|S| = d$ there exists $i \in M_1$ such that*

$$\sum_{j \in S} a_{ij} < b_i \quad (4.8)$$

then

$$\sum_{j \in N_1} x_j \geq d + 1 \quad (4.9)$$

is a valid inequality for $C(\mathbf{A})$.

Proof. Assume there exists $x^1 \in C(\mathbf{A})$ that does not satisfy (4.9). This means $\sum_{j \in N_1} x_j^2 < d + 1$ for an $x^2 \in S(\mathbf{A})$. This implies that $\sum_{j \in N_1} x_j^2 \leq d$ since x^2 is integer. This last sentence implies the existence of a set $S \subseteq N_1$ with $|S| = d$ containing the set of indices $\{j : x_j^2 = 1, j \in N_1\}$. We know that for such S there exists $i \in M_1$ for which (4.8) holds. So x^2 does not satisfy one of the inequalities (4.1), which is a contradiction. Q.E.D.

We give a sufficient condition for (4.9) to define a facet of $C(\mathbf{A})$ with the following proposition. We assume that $C(\mathbf{A})$ is full dimensional.

Proposition 2. *Given (4.9), which is a valid inequality obtained using Proposition 1. If for a $T \subseteq N_1$ with $|T| = d + 2$ there exists edges*

$$e_k \subseteq (T \cup N_2) / \{k\} \quad k \in T \quad (4.10)$$

$$e_k \subseteq (T \cup N_2) / \{k, l_k\} \quad \text{for a } l_k \in T, k \in N_2 \quad (4.11)$$

$$e_k \subseteq (T \cup N_2) \cup \{k\} \quad \text{with } |T \cap e_k| = d \quad k \in N_1 \setminus T \quad (4.12)$$

on $H_A(N, E)$ then (4.9) is a facet of $C(\mathbf{A})$.

Proof. (4.10) – (4.12) imply the existence of n affinely independent points $z^k, k =$

$1, 2, \dots, n$ in $S(\mathbf{A})$ having the j th entry

$$z_j^k = \begin{cases} 0 & j \in (N_1 \setminus T) \cup \{k\} \\ 1 & \text{otherwise} \end{cases} \quad k \in T$$

$$z_j^k = \begin{cases} 0 & j \in (N_1 \setminus T) \cup \{k, l_k\} \\ 1 & \text{otherwise} \end{cases} \quad k \in N_2$$

$$z_j^k = \begin{cases} 1 & j \in e_k \cup \{k\} \\ 0 & \text{otherwise} \end{cases} \quad k \in N_1 \setminus T$$
(4.13)

and satisfying (4.9) as equality. The affine independance of these n points can be proved as follows. When we subtract point $\mathbf{z}^1$ from the remaining points we see that components in N_1/T of $\mathbf{z}^k - \mathbf{z}^1$ for $k \in N_1/T$ form an identity matrix. Components in N_1/T of the remaining $\mathbf{z}^k - \mathbf{z}^1$ points are zero. Also, components in N_2 of $\mathbf{z}^k - \mathbf{z}^1$ for $k \in N_2$ form the negative of an identity matrix. Components in N_2 of the remaining $\mathbf{z}^k - \mathbf{z}^1$ points are zero. Finally, the components in T of $\mathbf{z}^k - \mathbf{z}^1$ for T form a column of 1's attached the negative of an identity matrix. Q.E.D.

4.3. Sequential Lifting

Conditions for the validity and facet definition of (4.9) may not be satisfied for the whole $\mathbf{A}$ matrix but can be satisfied for a nonempty subset of its columns $N_r \subset N$ namely $\mathbf{A}_M^{N_r}$. In that case we use the sequential lifting given in [51]. The lifted inequality for $C(\mathbf{A})$ preserves the properties that the original has for $C(\mathbf{A}_M^{N_r})$. In this section we assume that $C(\mathbf{A}_M^{N_r})$ is non empty and show that sequential lifting of an inequality that is valid for $C(\mathbf{A}_M^{N_r})$ guarantee the validity of the lifted inequality for $C(\mathbf{A})$.

Proposition 3. *Given (4.9) a valid inequality for $C(\mathbf{A}_M^{N_r})$ obtained by applying Proposition 1 to $\mathbf{A}_M^{N_r}$, let $N_3 = N/N_r$ and $\{j_1, j_2, \dots, j_{|N_3|}\}$ any ordering of elements of N_3 . Then, the inequality*

$$\sum_{i \in N_1} x_i + \sum_{l=1}^{|N_3|} \alpha_{j_l}^u x_{j_l} \geq d + 1$$
(4.14)

where $\alpha_{j_l}^u$ is an upper bound for the optimum solution of the mathematical program

$$\text{Max} \quad \alpha_{j_l} \tag{4.15}$$

s.t.

$$\sum_{i \in N_1} x_i + \sum_{t=1}^{l-1} \alpha_{j_t}^u x_{j_t} + \alpha_{j_l} = d + 1 \tag{4.16}$$

$$\sum_{j \in N_r} a_{ij} x_j + \sum_{t=1}^{l-1} a_{ij_t} x_{j_t} + a_{ij_l} \geq b_i \quad i \in M \tag{4.17}$$

$$x_j = \{0, 1\} \quad j \in N_r, \quad x_{j_t} = \{0, 1\} \quad t \in \{1, 2, \dots, l-1\}, \quad \alpha_{j_l} \text{ unrestricted} \tag{4.18}$$

(4.14) is a valid inequality for $C(\mathbf{A})$.

Proof. (4.15) – (4.18) has a feasible solution since $C(\mathbf{A}_M^{N_r})$ is nonempty and its optimal value is always bounded as a consequence of the first constraint. Hence, it has a finite optimal solution. Assume (4.14) is not a valid inequality for $C(\mathbf{A})$. This means there exists an $\mathbf{x}^* \in S(\mathbf{A})$ satisfying

$$\sum_{i \in N_1} x_i^* + \sum_{l=1}^{|N_3|} \alpha_{j_l}^u x_{j_l}^* < d + 1 \tag{4.19}$$

Case 1: There is $s \geq 1$ such that s is the largest integer $x_{j_s}^* = 1$ for all $s \in \{1, 2, \dots, |N_3|\}$.

This implies

$$\sum_{i \in N_1} x_i^* + \sum_{i=1}^{s-1} \alpha_{j_i}^u x_{j_i}^* + \alpha_{j_s}^u < d + 1, \tag{4.20}$$

and $(\mathbf{x}^{*t}, \alpha)$ is a feasible solution of the mathematical program defining $\alpha_{j_s}^u$ for an α that satisfies the first constraint (4.16) with $\mathbf{x}^{*t}$. $\mathbf{x}^{*t}$ is the point formed by truncating the components $\{j_s, j_{s+1}, \dots, j_{|N_3|}\}$ of $\mathbf{x}^*$. But this last sentence implies $\alpha_{j_s}^u \geq \alpha$. Therefore,

$$\alpha_{j_s}^u \geq d + 1 - \sum_{i \in N_1} x_i^* - \sum_{i=1}^{s-1} \alpha_{j_i}^u x_{j_i}^*, \tag{4.21}$$

which contradicts (4.20).

Case 2: There is no such s . This means $x_{j_t}^* = 0$ for all $t \in \{1, 2, \dots, |N_3|\}$ and $\mathbf{x}^* \in C(\mathbf{A}_M^{N_r})$. Remember that (4.9) is a valid inequality for $C(\mathbf{A}_M^{N_r})$. So $\sum_{i \in N_1} x_i^* + \sum_{l=1}^{|N_3|} \alpha_{j_l}^u x_{j_l}^* \geq d + 1$, which contradicts (4.19). Q.E.D.

It is also possible to show that we can obtain a facet defining inequality of $C(\mathbf{A})$ by sequentially lifting a facet defining inequality of $C(\mathbf{A}_M^{N_r})$.

Proposition 4. *Consider (4.9) which is a valid inequality for $C(\mathbf{A}_M^{N_r})$ obtained by applying Proposition 1 on $\mathbf{A}_M^{N_r}$ and assume $C(\mathbf{A}_M^{N_r})$ and $C(\mathbf{A})$ are full dimensional. If (4.9) defines a facet of $C(\mathbf{A}_M^{N_r})$ then (4.14) defines a facet of $C(\mathbf{A})$ with $\alpha_{j_l}^u$ being an exact optimal value of (4.15) – (4.18) .*

Proof. It suffices to show that there exists $|N_r|$ affinely independent points $\mathbf{x}^t$ $t = 1, 2, \dots, |N_r|$ of $S(\mathbf{A}_M^{N_r})$ such that

$$\sum_{i \in N_1} x_i^t = d + 1 \quad (4.22)$$

since (4.9) defines a facet of $C(\mathbf{A}_M^{N_r})$. Let us consider the following n points in $\mathbb{R}^n$

$$p_k^i = \begin{cases} x_k^i & k \in N_r \\ 0 & \text{otherwise} \\ y_k^{j_i} & k \in N_r \cup \{j_1, j_2, \dots, j_{i-1}\} \\ 1 & k = j_i \\ 0 & \text{otherwise} \end{cases} \quad \begin{matrix} i \in N_r \\ \\ i \in \{1, 2, \dots, |N_3|\} \end{matrix} \quad (4.23)$$

In (4.23) $(\mathbf{y}^{j_i}, \alpha_{j_i}^u)$ is an optimal solution of (4.15) – (4.18) giving $\alpha_{j_i}^u$. The integer points $\mathbf{p}^i$, $i = 1, 2, \dots, n$ are affinely independent because when we subtract $\mathbf{p}^1$ from the remaining points we see that components in N_3 of $\mathbf{p}^i - \mathbf{p}^1$ for $i \in \{1, 2, \dots, |N_3|\}$ form an identity matrix. The integer points $\mathbf{p}^i$, $i = 1, 2, \dots, n$ are element of $S(\mathbf{A})$ because $\mathbf{x}^t$ $t = 1, 2, \dots, |N_r|$ define a facet of $C(\mathbf{A}_M^{N_r})$ and $\mathbf{y}^{j_i}$ satisfies inequalities (4.17). The integer points $\mathbf{p}^i$, $i = 1, 2, \dots, n$ also satisfy (4.14) as equality by construction. Therefore,

(4.14) defines a facet of $C(\mathbf{A})$.

Q.E.D.

Recall that Proposition 1 assumes that a positive integer d is given. It is possible to determine a value for d by using the optimal value of the LP relaxation of the following integer program:

$$\text{Min} \quad \sum_{i \in N_1} x_i \quad (4.24)$$

s.t.

$$\sum_{j \in N_1} a_{ij} x_j \geq b_i \quad i \in M_1 \quad (4.25)$$

$$x_j = \{0, 1\} \quad j \in N_1. \quad (4.26)$$

The optimal value of (4.24) – (4.26), if it exists, corresponds to $d + 1$ in (4.9), which can be obtained by rounding the optimal objective value of its LP relaxation up to the nearest integer.

Proposition 3 shows that finding an upper bound on the optimal value of (4.15) – (4.18) is sufficient to obtain a valid inequality for $C(\mathbf{A})$. This can be done by solving its LP relaxation in polynomial time. Notice that for a facet we have to find the optimal solution.

Also, changing the lifting order $\{j_1, j_2, \dots, j_{|N_3|}\}$ in Proposition 3 can give a different valid inequality. Starting with a small cardinality N_r , we can find the positive integer d described in Proposition 1 using (4.24) – (4.26) and thus construct the valid inequality (4.9) for $C(\mathbf{A}_M^{N_r})$. $|N_r|$ should not be too small in order to obtain a large d , therefore a stronger valid inequality. We can then lift the obtained valid inequality using the optimum value of the LP relaxation of (4.15) – (4.18). We can repeat these steps with different orderings $\{j_1, j_2, \dots, j_{|N_3|}\}$ and change the initial N_r set to obtain different valid inequalities. The procedure is summarized in Figure 4.1.

Proposition 5. *The worst case time complexity of Algorithm in Figure 4.1 is $\mathcal{O}(T_1 T_2 n n_A^3 L)$ complexity where $n_A = \max(m, n)$, L is the length of binary bits required to store the*

-
1. For a predetermined number of times T_1 Do
 - 1.1. Choose a column set N_r which is not used before.
 - 1.2. Solve (4.24) – (4.26) to determine a positive integer d .
 - 1.3. Construct a valid inequality (4.9) for $C(\mathbf{A}_m^{N_r})$ with d
 - 1.4. For a predetermined number of times T_2 Do
 - 1.4.1. Consider a new ordering $\{j_1, j_2, \dots, j_{|N_3|}\}$
 - 1.4.2. Lift the inequality with N_3 by solving LP relaxation of (4.15) – (4.18) according to Proposition 3.
 - End For
 - End For
-

Figure 4.1. Valid inequality generation

data of an (4.15) – (4.18) instance, based on the time complexity of Karmarkar's algorithm [60].

Proof. The number of columns and rows in the LP are respectively $\mathcal{O}(n)$ and $\mathcal{O}(m)$. We solve the LP $T_1 T_2 n$ times in the worst case. Q.E.D.

4.4. Numerical Example: Valid Inequality

We choose the following symmetric $\mathbf{A}$ matrix and b vector from an instance of the probabilistic coverage problem given in [11].

$$\mathbf{A} = \begin{bmatrix} 6.908 & 1.609 & 0.511 & 1.609 & 1.204 & 0.357 \\ 1.609 & 6.908 & 1.609 & 1.204 & 1.609 & 1.204 \\ 0.511 & 1.609 & 6.908 & 0.357 & 1.204 & 1.609 \\ 1.609 & 1.204 & 0.357 & 6.908 & 1.609 & 0.511 \\ 1.204 & 1.609 & 1.204 & 1.609 & 6.908 & 1.609 \\ 0.357 & 1.204 & 1.609 & 0.511 & 1.609 & 6.908 \end{bmatrix} \quad b = \begin{bmatrix} 2.303 \\ 2.303 \\ 2.303 \\ 2.303 \\ 2.303 \\ 2.303 \end{bmatrix} \quad (4.27)$$

It is easy to realize that for $N_r = \{1, 3, 4, 5, 6\}$, $N_1 = N_r$, $N_2 = \emptyset$, $M_1 = M$ one can find a constraint row that is not satisfied by any selected two columns. $C(\mathbf{A}_M^{N_r})$ is not empty since $\mathbf{1}$ satisfies (4.1) for $\mathbf{A}_M^{N_r}$. Therefore by Proposition 1

$$\sum_{\substack{i=1 \\ i \neq 2}}^6 x_i \geq 3 \quad (4.28)$$

is a valid inequality for $C(\mathbf{A}_M^{N_r})$. Choosing T as $\{1, 3, 4, 6\}$ in Proposition 2 $e_1 = \{3, 4, 6\}$, $e_3 = \{1, 4, 6\}$, $e_4 = \{1, 3, 6\}$, $e_6 = \{1, 3, 4\}$, $e_5 = \{1, 3, 5\}$, are edges on $H_{A_M^{N_r}}(N_r, E)$ being subsets of $T/\{1\}$, $T/\{3\}$, $T/\{4\}$, $T/\{6\}$, $T \cup \{5\}$, respectively. Note that $|T \cap e_5| = d = 2$ and the condition of Proposition 2 is satisfied. Therefore by Proposition 2 (4.28) defines a facet of $C(\mathbf{A}_M^{N_r})$. Notice that the vectors with components corresponding to columns 1,3,4,6,5

$$\mathbf{z}^1 = (0, 1, 1, 1, 0)^T$$

$$\mathbf{z}^3 = (1, 0, 1, 1, 0)^T$$

$$\mathbf{z}^4 = (1, 1, 0, 1, 0)^T$$

$$\mathbf{z}^6 = (1, 1, 1, 0, 0)^T$$

$$\mathbf{z}^5 = (1, 1, 0, 0, 1)^T$$

are affinely independent because as can be checked easily the vectors $\mathbf{z}^3 - \mathbf{z}^1, \mathbf{z}^4 - \mathbf{z}^1, \mathbf{z}^6 - \mathbf{z}^1$ and $\mathbf{z}^5 - \mathbf{z}^1$ are linearly independent.

Now let us apply Proposition 4 to obtain a facet defining valid inequality for $C(\mathbf{A})$. $C(\mathbf{A}_M^{N_r})$ is full dimensional since the points t^i $1 \leq i \leq 5$ with all components equal to one except i^{th} being zero satisfy (4.1) for constraint matrix $\mathbf{A}_M^{N_r}$. By a similar argument it is easy to see that $C(\mathbf{A})$ is also full dimensional. $N_3 = \{2\}$. As it is

explained in Proposition 4 we have to optimally solve (4.15) – (4.18) namely

$$\text{Max } \alpha_2 \tag{4.29}$$

s.t.

$$x_1 + x_3 + x_4 + x_5 + x_6 + \alpha_2 = 3 \tag{4.30}$$

$$6.908x_1 + 0.511x_3 + 1.609x_4 + 1.204x_5 + 0.357x_6 \geq 0.694 \tag{4.31}$$

$$1.609x_1 + 1.609x_3 + 1.204x_4 + 1.609x_5 + 1.204x_6 \geq -4.605 \tag{4.32}$$

$$0.511x_1 + 6.908x_3 + 0.357x_4 + 1.204x_5 + 1.609x_6 \geq 0.694 \tag{4.33}$$

$$1.609x_1 + 0.357x_3 + 6.908x_4 + 1.609x_5 + 0.511x_6 \geq 1.099 \tag{4.34}$$

$$1.204x_1 + 1.204x_3 + 1.609x_4 + 6.908x_5 + 1.609x_6 \geq 0.694 \tag{4.35}$$

$$0.357x_1 + 1.609x_3 + 0.511x_4 + 1.609x_5 + 6.908x_6 \geq 1.099 \tag{4.36}$$

$$x_j = \{0, 1\} \ j \in \{1, 3, 4, 5, 6\} \ \alpha_2 \text{ unrestricted} \tag{4.37}$$

to obtain a facet defining valid inequality. Observe that when $x_5 = 1$ all the constraints are satisfied and $\alpha_2 = 2$ is an optimal solution. Therefore

$$\sum_{\substack{i=1 \\ i \neq 2}}^6 x_i + 2x_2 \geq 3 \tag{4.38}$$

is a facet defining valid inequality for $C(\mathbf{A})$. Choosing the x^t vectors as z^1, z^3, z^4, z^6, z^5 and using $y^{j_1} = (0, 0, 0, 0, 1)^T$ we obtain

$$\mathbf{p}^1 = (0, 1, 1, 1, 0, 0)^T$$

$$\mathbf{p}^3 = (1, 0, 1, 1, 0, 0)^T$$

$$\mathbf{p}^4 = (1, 1, 0, 1, 0, 0)^T$$

$$\mathbf{p}^6 = (1, 1, 1, 0, 0, 0)^T$$

$$\mathbf{p}^5 = (1, 1, 0, 0, 1, 0)^T$$

$$\mathbf{p}^2 = (0, 0, 0, 0, 1, 1)^T.$$

These are affinely independent because as can be checked easily the vectors $p^3 - p^1, p^4 - p^1, p^6 - p^1, p^5 - p^1$ and $p^2 - p^1$ are linearly independent.

It is worth to note that (4.38) cuts off a fractional vertex of $C(\mathbf{A})$, namely $\mathbf{x}^* = (0.205, 0.135, 0.205, 0.205, 0.135, 0.205)^T$. Hence, it is a strong valid inequality. $\mathbf{x}^*$ is a vertex of $C(\mathbf{A})$ because it satisfies (4.1) as equality and $\mathbf{A}$ is a matrix of full rank.

If we apply Algorithm in Figure 4.1 on this example with only one column set $N_r = \{1, 3, 4, 5, 6\}$ and one N_3 ordering $\{2\}$, after determining that (4.28) is a valid inequality, we simply solve the LP relaxation of (4.29) – (4.37). The result is $\alpha_2 = 2.61$; but when we look at (4.29) – (4.37) we realize that in its exact integer solution, α_2 has to be an integer. Therefore $\alpha_2 = 2$ is also an upper bound for (4.29) – (4.37). We again obtain (4.38) as a valid inequality for $C(\mathbf{A})$.

4.5. A Cutting Plane for the Binary Covering Integer Programs

In this section we propose a cutting plane for $S(\mathbf{A})$. The following variant of Farkas' Lemma [61] is crucial for its theoretical foundation.

Proposition 6. *Given an n dimensional nonnegative vector $\boldsymbol{\pi}$, a matrix $\mathbf{G}$ having p rows and n columns, and a number π_0 one and only one of the following systems has a solution*

$$\begin{array}{ll} \boldsymbol{\pi}^T \mathbf{x} < \pi_0 & \mathbf{v}^T \mathbf{1} \geq \pi_0 \\ S_1 : \mathbf{G}\mathbf{x} \geq \mathbf{1} & S_2 : \mathbf{G}^T \mathbf{v} \leq \boldsymbol{\pi} \\ \mathbf{x} \geq \mathbf{0} & \mathbf{v} \geq \mathbf{0} \end{array} \quad (4.39)$$

$S(\mathbf{A})$ can equivalently be written as $S(\mathbf{A}) = P(\mathbf{A}) \cap \mathbb{Z}^n$ with

$$P(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{F}\mathbf{x} \geq \mathbf{f}, \mathbf{x} \geq \mathbf{0}\} \quad (4.40)$$

where $\mathbf{F} = \begin{bmatrix} \mathbf{A} \\ -\mathbf{I}^n \end{bmatrix}$ and $\mathbf{f} = \begin{bmatrix} \mathbf{b} \\ -1 \end{bmatrix}$. Then, we can show the following proposition.

Proposition 7. *Let $\mathbf{u} \geq 0$ be an $m + n$ dimensional vector satisfying*

$$\mathbf{s}^T = \mathbf{c}^T - \mathbf{u}^T \mathbf{F} \geq \mathbf{0}^T, \quad (4.41)$$

and $\mathbf{G}$ be a matrix having p rows and n columns with $g_{ij} = 0$ or 1 . If there exists a p dimensional $\mathbf{v}$ vector such that

$$\mathbf{v}^T \mathbf{1} \geq z^u - \mathbf{u}^T \mathbf{f} \quad (4.42)$$

$$\mathbf{G}^T \mathbf{v} \leq \mathbf{s} \quad (4.43)$$

$$\mathbf{v} \geq \mathbf{0}, \quad (4.44)$$

then all $x \in S(\mathbf{A})$ with a smaller objective value for problem IPC than z^u , i.e. $\mathbf{c}^T \mathbf{x} < z^u$, satisfies the disjunction

$$\bigvee_{i=1}^p (x_j = 0, \ j : g_{ij} = 1). \quad (4.45)$$

Proof. For all $\mathbf{x} \in S(\mathbf{A})$

$$\mathbf{F} \mathbf{x} \geq \mathbf{f} \quad (4.46)$$

by definition of $P(\mathbf{A})$. The facts that $\mathbf{u} \geq 0$ and $\mathbf{c}^T \mathbf{x} < z^u$ together with (4.46) imply

$$(\mathbf{c}^T - \mathbf{u}^T \mathbf{F}) \mathbf{x} < z^u - \mathbf{u}^T \mathbf{f}. \quad (4.47)$$

Defining

$$\boldsymbol{\pi}^T = \mathbf{s}^T = (\mathbf{c}^T - \mathbf{u}^T \mathbf{F}) \quad (4.48)$$

$$\pi_0 = z^u - \mathbf{u}^T \mathbf{f} \quad (4.49)$$

we see that (4.42) – (4.44) imply the existence of a $\mathbf{v}$ vector satisfying S_2 in (4.39). By Proposition 6 there is no vector $\mathbf{x} \geq 0$ satisfying (4.47) and

$$\mathbf{G}\mathbf{x} \geq \mathbf{1} \quad (4.50)$$

at the same time. This implies the disjunction in (4.45).

Q.E.D.

We obtain a disjunctive inequality from the disjunction in Proposition 7 as follows. We first divide each row $i \in M$ of matrix $\mathbf{A}$ by b_i (i.e. the i th entry of $\mathbf{b}$) in (4.1) and obtain the equivalent inequality set

$$\mathbf{A}^1 \mathbf{x} \geq \mathbf{1}. \quad (4.51)$$

where $\mathbf{A}^1$ is the matrix that comes out from the described scaling operation. We then find a feasible solution to problem IPC using a greedy heuristic described in Figure 4.3 at the end of the section. The objective value of the feasible solution is the upper bound z^u used in Proposition 7. The next step is the solution of LPC . We then form $\mathbf{G}$ matrix described in Proposition 7 using the following rule:

$$g_{ij} = \begin{cases} 1 & \text{if } x_j^{lpc} \geq p_1, a_{ij} \geq p_2 \\ 0 & \text{otherwise} \end{cases} \quad i = 1, 2, \dots, m; \quad j = 1, 2, \dots, n. \quad (4.52)$$

In (4.52) p_1 and p_2 are two nonnegative real parameters where $p_1 \leq 1$. When they have low values the cutting plane gets stronger but there is a danger of not being able to satisfy conditions (4.42) – (4.44) of Proposition 7. When the values are too high,

the cutting plane gets weaker. After determining matrix G we solve the LP

$$\min \quad \mathbf{v}^T \mathbf{1} \quad (4.53)$$

s.t.

$$\mathbf{v}^T \mathbf{1} + \mathbf{u}^T \mathbf{f} \geq z^u \quad (4.54)$$

$$\mathbf{G}^T \mathbf{v} \leq \mathbf{s} \quad (4.55)$$

$$\mathbf{u}^T \mathbf{F} + \mathbf{s}^T = \mathbf{c}^T. \quad (4.56)$$

The nonnegative continuous decision variables $\mathbf{u}, \mathbf{v}, \mathbf{s}$ are as described in Proposition 7. The objective function (4.53) minimizes the sum of the components of $\mathbf{v}$. Constraints (4.54) – (4.56) are simply the conditions of Proposition 7. Therefore, the LP (4.53) – (4.56) gives the $\mathbf{v}$ vector having the smallest component sum among the $\mathbf{u}, \mathbf{v}, \mathbf{s}$ vectors satisfying the conditions of Proposition 7. Notice that the number of rows used are decreased and cutting plane becomes stronger as long as the sum of the components of $\mathbf{v}$ vector becomes smaller. In forming the cutting plane, each row i of the $\mathbf{G}$ matrix is matched with row i of $\mathbf{A}^1$ matrix if v_i is greater zero.

By Proposition 7, the existence of a solution $\mathbf{u}^*, \mathbf{v}^*, \mathbf{s}^*$ of the LP (4.53) – (4.56) implies the following disjunction

$$\bigvee_{i=1}^m (x_j = 0, \quad j : g_{ij} = 1) \quad (4.57)$$

for $x \in S(\mathbf{A})$ with $\mathbf{c}^T \mathbf{x} < z^u$. In addition, after matching each row i of $\mathbf{G}$ with row i of $\mathbf{A}^1$ and using the fact that any $\mathbf{x} \in S(\mathbf{A})$ with a smaller objective function value for problem *IPC* than z^u satisfies (4.51) and (4.57) we obtain the following disjunction

$$\bigvee_{i=1}^m \left\{ \sum_{j=1}^n a_{ij}^1 (1 - g_{ij}) x_j \geq 1 \right\}. \quad (4.58)$$

Before proceeding further in the derivation of a cutting plane we try to simplify (4.58). If $v_l^* = 0$ for a $l \in \{1, 2, \dots, m\}$, a reduced vector $\mathbf{v}^{*1}$ of dimension $m - 1$ that does

not have old l^{th} component, with a reduced $\mathbf{G}^1$ matrix of dimension $(m-1) \times n$ that does not have old l^{th} row and with old $\mathbf{u}^*, \mathbf{s}^*$ satisfy the conditions of Proposition 7. Therefore, we can exclude the part of the disjunction that corresponds to $i = l$ in (4.58). Moreover, if $\sum_{j=1}^n \{a_{ij}^1 (1 - g_{ij})\}$ is less than 1 for any part i of the disjunction (4.58) then we can also exclude that part because it cannot be satisfied. After these simplifications we obtain the following disjunction

$$\bigvee_{i=1}^t \left\{ \sum_{j=1}^n a_{ij}^1 (1 - g_{ij}) x_j \geq 1 \right\}. \quad (4.59)$$

In (4.59) t is an integer smaller or equal to m . We assume without loss of generality that the excluded disjunction parts are from $t+1$ to m . Then, for i in the disjunction we find

$$j_i^* = \arg \max_{1 \leq j \leq n} \left\{ \min [a_{ij}^1 (1 - g_{ij}), 1] \right\}. \quad (4.60)$$

If

$$maxval_i = \min [a_{ij_i^*}^1 (1 - g_{ij_i^*}), 1] \quad (4.61)$$

is strictly less than one then we add i to the set $rhsset$ that determines the right hand side coefficient c_{rhs} of the cutting plane. c_{rhs} is then found as

$$c_{rhs} = \min_{i \in rhsset} \left\lceil \frac{1}{maxval_i} \right\rceil. \quad (4.62)$$

If $rhsset$ is empty then $c_{rhs} = 1$. Using (4.59) and the fact that elements of $S(\mathbf{A})$ that give a better objective value than z^u are integer points we can write

$$\bigvee_{i \in rhsset} \left\{ \sum_{j=1}^n \left\lceil \frac{a_{ij}^1 (1 - g_{ij})}{maxval_i} \right\rceil x_j \geq c_{rhs} \right\} \bigvee_{i \notin rhsset} \left\{ \sum_{j=1}^n a_{ij}^1 c_{rhs} (1 - g_{ij}) x_j \geq c_{rhs} \right\}. \quad (4.63)$$

This implies the following inequality

$$\sum_{j=1}^n \left\{ \max \left(\max_{\substack{i \in rhsset \\ i \in \{1,2,\dots,t\}}} \left\lceil \frac{a_{ij}^1 (1 - g_{ij})}{maxval_i} \right\rceil, \max_{\substack{i \notin rhsset \\ i \in \{1,2,\dots,t\}}} a_{ij}^1 c_{rhs} (1 - g_{ij}) \right) \right\} x_j \geq c_{rhs} \quad (4.64)$$

that is satisfied by all $x \in S(\mathbf{A})$ that give a better objective value than z^u . The procedure of obtaining the cutting plane (4.64) is summarized in Figure 4.2.

Proposition 8. *The worst case complexity of Algorithm in Figure 4.2 is $\mathcal{O}((m+n)^3 L_v + n_A L_{LPC} + n^2 m)$, where L_{LPC} denotes the length of binary bits required to store an LPC instance, L_v denotes the length of binary bits required to store an instance of the LP (4.53) – (4.56).*

Proof. The largest of the number of constraints and the number of variables in LP (4.53) – (4.56) is $m+n$. The largest of the number of constraints and the number of variables in LPC is n_A . Then the worst case time is $\mathcal{O}(n^2 m)$ assuming that Karmarkar's algorithm [60] is used to solve the LP Q.E.D.

Problem *LPC* when solved after adding cutting plane (4.64) can give an optimal value greater than z^u . This implies that $\mathbf{x}^u \in S(\mathbf{A})$ such that $z^u = \mathbf{c}^T \mathbf{x}^u$ is the optimal value as the following proposition shows.

Proposition 9. *If problem *LPC* gives optimal value lb when solved after adding the cutting plane (4.64) to the constraint set and if*

$$lb \geq z^u, \quad (4.65)$$

then $\mathbf{x}^u \in S(\mathbf{A})$ such that $z^u = \mathbf{c}^T \mathbf{x}^u$ is the optimal value.

Proof. Suppose there exists an $\mathbf{x}^1 \in S(\mathbf{A})$ with a better objective value i.e.,

$$\mathbf{c}^T \mathbf{x}^1 < z^u \quad (4.66)$$

-
1. Initialization: Assign values to parameters p_1, p_2 . Initialize set $rhsset$ to $\emptyset$
 2. For $1 \leq i \leq m$ Do
 - 2.1. Divide the i^{th} row a_i of $\mathbf{A}$ by b_i to have the equivalent form (4.51) for (4.1)
 End For
 3. Find a feasible solution to problem IPC using Algorithm in Figure (4.3)
 4. Assign the corresponding objective value to z^u
 5. Solve LPC and record an optimal solution as x^{lpc}
 6. Form G matrix using the rule described in (4.52)
 7. Solve the LP (4.53) – (4.56)
 8. For $1 \leq i \leq m$ Do
 - 8.1. Exclude part i from the disjunction if $v_i^* = 0$ or $\sum_{j=1}^n \{a_{ij}^1 (1 - g_{ij})\}$ is strictly less than 1
 End For
 9. For $1 \leq i \leq t$ Do
 - 9.1. Find j_i^* using (4.60)
 - 9.2. If $maxval_i$ given by (4.61) is strictly less than one add i to $rhsset$
 End For
 10. If $rhsset$ is $\emptyset$ then set $c_{rhs} = 1$

Else calculate c_{rhs} from (4.62)

 Endif
 11. Form the cutting plane using (4.64)
-

Figure 4.2. Construction of a cutting plane

then from Proposition 7 $\mathbf{x}^1$ satisfies cutting plane condition (4.64). Therefore, $\mathbf{x}^1$ is a feasible solution of the linear program giving an objective value equal to lb which is described in the statement of the proposition. This implies

$$\mathbf{c}^T \mathbf{x}^1 \geq lb, \quad (4.67)$$

from which

$$lb < z^u \quad (4.68)$$

follows. This clearly contradicts (4.65), which completes the proof.

Q.E.D.

Finally we describe the greedy heuristic used in finding z^u in the algorithm of Figure 4.2. The algorithm assumes that constraints (4.1) are transformed into the equivalent constraints (4.51). In the algorithm, a column $j \in N$ is included into the set of selected columns sc at each iteration. Let $brem$ be an m dimensional vector with

$$brem_i = \max \left[\left(1 - \sum_{j \in sc} a_{ij} \right), 0 \right] \quad i \in M. \quad (4.69)$$

This means $brem$ vector stores the quantity to be added to the left hand side of constraint $i \in M$ in (4.51) by selecting more columns so that it is larger than or equal to one. At each iteration, first $brem$ vector is calculated. If all of its components are zero the algorithm terminates. Vector $\mathbf{x}^u$ with

$$x_j^u = \begin{cases} 1 & \text{if } j \in sc \\ 0 & \text{otherwise} \end{cases} \quad (4.70)$$

is the solution found by the algorithm. Otherwise, cost satisfaction ratio $ctst_j$ is calculated for all j element of N not element of sc with a positive $\sum_{i \in M} \min(a_{ij}, brem_i)$ value. The expression for $ctst_j$ is

$$ctst_j = \frac{c_j}{\sum_{i \in M} \min(a_{ij}, brem_i)}. \quad (4.71)$$

If there is no such j then there is a problem with $\mathbf{A}^1$ matrix, right hand side vector $\mathbf{1}$ in (4.51) can not be satisfied even if all columns are selected. The algorithm stops. Otherwise column $j \notin sc$ having the least $ctst_j$ ratio is added to sc . $brem$ vector is updated and another iteration is performed. The overall procedure is summarized in

Figure 4.3.

-
1. Initialization: sc is $\emptyset$. $brem_i = 1 \ i = 1, 2, \dots, m$
 2. While $\left(\sum_{i \in M} brem_i > 0 \right)$ Do
 - 2.1. Calculate $ctst_j$ from (4.71) for all $j \notin sc$ with positive $\sum_{i \in M} \min(a_{ij}, brem_i)$ value
 - 2.2.1. If $\left(\sum_{i \in M} \min(a_{ij}, brem_i) \right) = 0$ for all $j \notin sc$ Then stop. There is a problem with $\mathbf{A}^1$ matrix. The problem is infeasible
 - Else
 - 2.2.2. Find $j^* = \underset{j \notin sc}{\operatorname{argmin}} ctst_j$
 - 2.2.3. Add j^* to sc
 - 2.2.4. Update $brem$ using (4.69)
 - End If
 - End While
 3. Vector x^u given by (4.70) is the output
-

Figure 4.3. Greedy heuristic

4.6. Numerical Example: Cutting Plane

To illustrate the cutting plane generation algorithm given in Figure 4.2 we present two numerical examples. In the first one the following instance of problem IPC with

$$\mathbf{A} = \begin{bmatrix} 1.366 & 0 & 0.903 & 0 & 0 \\ 0 & 1.048 & 0 & 0.719 & 0 \\ 1.186 & 0 & 0.589 & 0 & 1.275 \end{bmatrix},$$

$\mathbf{b}$ vector is $\mathbf{1}$ and $\mathbf{c}^T = (1, 1, 2, 2, 3)$ is considered. Parameter p_1 is set to 0 and p_2 is set to 1. Problem LPC when solved with CPLEX 11.0 [62] gives the solution $\mathbf{x}^{lpc} = (0.843, 0.954, 0, 0, 0)^T$ with 1.797 as the objective value. Greedy heuristic computes

$\mathbf{x}^u = (1, 1, 0, 0, 0)^T$, with $z^u = 2$. The rule described in (4.52) produces

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Then the solution of the LP (4.53) – (4.56) results in the optimal vectors $\mathbf{v}^* = [0, 1, 1]$, $\mathbf{u}^* = [0, 0, 0, 0, 0, 0, 0, 0]$ and $\mathbf{s}^* = [1, 1, 2, 2, 3]$. We can exclude disjunction parts 1 and 2 since $v_1^* = 0$ and $\sum_{j=1}^n \{a_{2j}^1 (1 - g_{2j})\}$ is less than 1. $\maxval_3 = 0.589$ and $rhsset = \{3\}$. From (4.62) c_{rhs} is 2. From (4.64) we obtain the following inequality satisfied by all $\mathbf{x} \in S(\mathbf{A})$ with a better objective value than 2

$$x_3 + 3x_5 \geq 2. \quad (4.72)$$

Problem *LPC* when solved after adding (4.72) to the constraint set gives 3.686 as optimal value. Therefore, there is no $\mathbf{x} \in S(\mathbf{A})$ with a better objective function value than 2 as a consequence of Proposition 9 and $\mathbf{x}^u = (1, 1, 0, 0, 0)^T$ is optimal.

Another instance of problem *IPC* has the following $\mathbf{A}$ matrix,

$$\mathbf{A} = \begin{bmatrix} 0.760 & 0 & 1.086 & 0 & 0 \\ 0 & 0.653 & 0 & 0.641 & 0 \\ 1.026 & 0 & 1.222 & 0 & 1.317 \end{bmatrix}.$$

The b and c vectors are kept the same as in the previous example. Parameter p_1 and p_2 are respectively set to 0 and 1 again. When problem *LPC* is solved with CPLEX 11.0 [62] we find the optimum solution $\mathbf{x}^{lpc} = (1, 1, 0.221, 0.541, 0)^T$ with 3.525 as the objective value. Greedy heuristic gives $\mathbf{x}^u = (1, 1, 1, 1, 0)^T$ with $z^u = 6$ and

$$x_2 + x_4 \geq 2 \quad (4.73)$$

is constructed as the new cutting plane. It is satisfied by all $\mathbf{x} \in S(\mathbf{A})$ with an objective

value better than 6. Problem LPC when solved after adding (4.73) to the constraint set gives 4.442 as the objective value. Inequality (4.73) cut off $\mathbf{x}^{lpc}$ and improve the lower bound given by LPC 26.01%.

5. LOCATION, SCHEDULING AND ROUTING PROBLEM

5.1. Introduction

In this chapter we first present a MILP formulation for the Location, Scheduling and Routing Problem (LSRP). The aim of LSRP is to determine the optimal location for the sensors and sinks, to find activity schedules of the sensors and sensor-to-sink data flow routes in a way to maximize the network lifetime subject to coverage, flow balance, energy and budget constraints.

5.2. Model Formulation

We consider a sensor field consisting of J points indexed as $j = 1, \dots, J$. The set of potential sensor and sink locations consists of the same J points. In the sequel we use both i and j as indices for sensor locations. We assume that there are K different types of sensors indexed as $k = 1, \dots, K$ with known sensing (r_k^s) and communication (r_k^c) ranges. We do not necessarily assume that $r_k^c \geq 2r_k^s$ for $k = 1, \dots, K$, as it is done in almost all of the works on homogeneous WSNs. It has been shown that coverage implies connectivity in a homogeneous WSN if this condition holds [63]. However, this is not always true for heterogeneous WSNs. A counter example is provided in Appendix A. We assume that a sensor of type- k placed at point i (referred to as sensor (i, k) in the sequel) generates h_{ik} data packets (unit that measures the volume of data) per period and that the following energy consumption rates per period are given for each type- k sensor in active mode: e_{ik}^s units of energy for sensing and data processing at point i during a period, e_k^r units of energy for receiving one data packet, and e_k^c units of energy for transmitting one data packet. The initial battery energy E_k and the unit cost c_{ik} of placing a type- k sensor on point i are also known. There is a total available budget of B monetary units and the number of sinks to be located is given as S . We assume that there is an upper bound on the amount of data sent from a type- k sensor to a

type- l sensor, which is denoted by u_{kl} . A similar type of upper bound exists also for the direct flow from a type- k sensor to a sink, which is given by v_k . Since the objective of the model is the maximization of the coverage lifetime L of the WSN, we assume a sufficiently long planning horizon that consists of T periods of equal durations. In other words, T becomes an upper bound on L . In each period $t = 1, \dots, T$, a sensor is either in active mode or in standby mode, and the energy expenditure of a sensor in standby mode is negligible, which can be justified easily by the technical specifications of the existing commercial sensors (xbow [46]). We also assume that the energy saved while a sensor is in standby mode is higher than the instantaneous energy required for activating the sensor. This is a valid assumption as long as the periods are not of exceptionally short durations. Finally, sinks have unlimited battery energy and remain in operation during the planning horizon.

In the following we introduce a MILP model that locates the sensors and sinks, and determines the activity schedule of the sensors, sensor-to-sensor and sensor-to-sink data flow routes maximizing WSN's lifetime. We use seven sets of decision variables. The first two are used in the placement of the sensors and the determination of their activity schedules: $x_{ik} = 1$ if a type- k sensor is placed at point i of the sensor field, $x_{ik} = 0$ otherwise; $z_{ikt} = 1$ if sensor (i, k) is active during period t , $z_{ikt} = 0$ otherwise. Third set of binary variables is used to locate the sinks: $s_i = 1$ if a sink is placed at site i , $s_i = 0$ otherwise. The last one is to determine whether or not period t is within the lifetime of the WSN: $y_t = 1$ if period t is within the lifetime L , $y_t = 0$ otherwise. In addition there are two sets of flow variables. The sensor-to-sensor flow variable f_{ikjlt} denotes the amount of data flow from sensor (i, k) to sensor (j, l) in period t . The sensor-to-sink flow variable g_{ikjt} denotes the amount of data flow from sensor (i, k) to a sink placed at point j in period t . Finally the single variable L represents the lifetime of the WSN, which is to be maximized.

We let a_{ijk} denote the nonnegative coverage coefficients. Their value is determined depending on the sensing characteristics (i.e. perfect, imperfect and probabilistic sensing). For perfect sensing, $a_{ijk} = 1$ if sensor (i, k) can cover point j , otherwise $a_{ijk} = 0$. For imperfect and uncertain sensing, in which sensor (i, k) may not perform its task

for point j with certainty, these parameters have nonnegative fractional values as explained in the recent work by Altinel et al. [12]. The right hand sides d_j of the coverage constraints are the given coverage quality requirements. Their value also depends on the sensing characteristics and the type of the coverage. They are positive integers for perfect sensing and become nonnegative rational numbers for imperfect and probabilistic sensing. They have distinct values for differentiated coverage. These details are also explained in the same work. The definition of the communication coefficients b_{ijk} is close to that of the coverage coefficients. In perfect communication: $b_{ijk} = 1$ if point j is within the transmission range of sensor (i, k) , otherwise $b_{ijk} = 0$. They must be modified properly after introducing imperfect and probabilistic transmission as new communication characteristics. Table 5.1 and Table 5.2 include a summary of all the decision variables and parameters used in the model.

Table 5.1. Decision variables used in the model

Decision variables	Definition
x_{ik}	One if a type- k sensor is placed at point i , zero otherwise
z_{ikt}	One if a type- k sensor located at point i is active in period t , zero otherwise
s_i	One if a sink is placed at point i , zero otherwise
y_t	One if period t is within the lifetime L , zero otherwise
f_{ikjlt}	Amount of data flow from sensor (i, k) to sensor (j, l) in period t
g_{ikjt}	Amount of data flow from sensor (i, k) to a sink placed at point j in period t
L	Lifetime of the WSN

Table 5.2. Parameters used in the model

Parameters	Definition
J	Number of points in the sensor field
K	Number of types of sensors
r_k^s	Sensing range of a type- k sensor
r_k^c	Communication range of a type- k sensor
h_{ik}	Number of data packets generated by a type- k sensor located at point i
e_{ik}^s	Energy consumption a type- k sensor located at point i for sensing and processing during a period
e_k^r	Energy consumption of a type- k sensor for receiving a data packet
e_k^c	Energy consumption of a type- k sensor for transmitting a data packet
E_k	Initial battery energy of a type- k sensor
c_{jk}	Cost of placing a type- k sensor at point j
B	Total available budget
S	Number of sinks to be located
u_{kl}	Upper bound on the amount of data sent from a type- k sensor to a type- l sensor
v_k	Upper bound on the amount of data sent from a type- k sensor to a sink
a_{ijk}	One if a type- k sensor located at point i can cover point j , zero otherwise
b_{ijk}	One if point j is within the transmission range of a type- k sensor located at point i , zero otherwise
d_j	Coverage quality requirement for point j
T	Planning horizon

Finally the MILP formulation of LSRP can be given as follows:

LSRP:

$$\max \quad L \quad (5.1)$$

s.t.

$$L \geq 1 \quad (5.2)$$

$$\sum_{i=1}^J \sum_{k=1}^K a_{ijk} z_{ikt} \geq d_j y_t \quad \text{for all } j, t \quad (5.3)$$

$$T y_t \geq L + 1 - t \quad \text{for all } t \quad (5.4)$$

$$h_{ik} z_{ikt} + \sum_{\substack{j=1 \\ j \neq i}}^J \sum_{l=1}^K b_{jil} f_{jlik t} = \sum_{\substack{j=1 \\ j \neq i}}^J \sum_{l=1}^K b_{ijk} f_{ikjlt} + \sum_{j=1}^J b_{ijk} g_{ikjt} \quad \text{for all } i, k, t \quad (5.5)$$

$$\sum_{t=1}^T \left(e_{ik}^s z_{ikt} + e_k^r \sum_{\substack{j=1 \\ j \neq i}}^J \sum_{l=1}^K b_{jil} f_{jlik t} + \sum_{\substack{j=1 \\ j \neq i}}^J e_k^c \sum_{l=1}^K b_{ijk} f_{ikjlt} + \sum_{j=1}^J e_k^c b_{ijk} g_{ikjt} \right) \leq E_k \quad \text{for all } i, k \quad (5.6)$$

$$z_{ikt} \leq x_{ik} \quad \text{for all } i, k, t \quad (5.7)$$

$$\sum_{k=1}^K \sum_{i=1}^J c_{ik} x_{ik} \leq B \quad (5.8)$$

$$\sum_{i=1}^J s_i = S \quad (5.9)$$

$$f_{ikjlt} \leq u_{kl} b_{ijk} z_{ikt} \quad \text{for all } i, k, j \neq i, l, t \quad (5.10)$$

$$f_{ikjlt} \leq u_{kl} b_{ijk} z_{jlt} \quad \text{for all } i, k, j \neq i, l, t \quad (5.11)$$

$$g_{ikjt} \leq v_k b_{ijk} z_{ikt} \quad \text{for all } i, k, j, t \quad (5.12)$$

$$g_{ikjt} \leq v_k b_{ijk} s_j \quad \text{for all } i, k, j, t \quad (5.13)$$

$$x_{ik}, z_{ikt}, s_i, y_t \in \{0, 1\}, f_{ikjlt} \geq 0, g_{ikjt} \geq 0. \quad (5.14)$$

Objective function (5.1) maximizes coverage lifetime. Constraint (5.2) puts a trivial bound on the coverage lifetime. Setting the lower bound as $L \geq 1$ removes from the feasible solution set the trivial solution $\{\mathbf{x} = \mathbf{0}, \mathbf{z} = \mathbf{0}, \mathbf{f} = \mathbf{0}, \mathbf{g} = \mathbf{0}, \mathbf{y} = \mathbf{0}\}$

resulting in $L = 0$, which is not meaningful since a realistic WSN should operate at least one period. Constraints (5.3) and (5.4) ensure that during the lifetime of the WSN (i.e. for $t \leq L$) coverage quality requirement d_j is satisfied at point j in the sensor field. Note that time is beyond the lifetime (i.e. $t > L$) when $y_t = 0$ and coverage constraint (5.3) becomes redundant. Constraints (5.5) are the data flow balance constraints. They ensure that if sensor (i, k) is active in period t , total data inflow including the data it generates is equal to the total data outflow. Note that transmission range limitation in receiving and sending data is taken into consideration in the determination of the communication coefficients b_{ijk} . Constraints (5.6) model energy restrictions. They ensure that total energy spent by sensor (i, k) , which is the sum of energy consumed in sensing and processing, energy spent in receiving and transmitting data during active time periods, is limited by the initial available battery energy. Constraint (5.7) guarantees that sensor (i, k) is active in period t only if it is placed. Constraint (5.8) ensures that the total deployment cost does not exceed total available budget. Constraint (5.9) sets the number of sinks to be placed. Constraints (5.10)–(5.11) enforce both sensor (i, k) and sensor (j, l) to be active during period t as a condition of data flow between them. They also put capacity restrictions on the amount of flows. Similarly constraints (5.12)–(5.13) do not allow flow from sensor (i, k) to a sink at point j if either the sensor is not active in period t or there is no sink located at point j . They put capacity restrictions on the amount of flows as well. We do not complicate the model by putting constraints limiting the number of active sensors at point i in period t . This is based on the observation that in the optimal solution, two sensors are rarely active at the same point within the same period, because this is not usually an energy efficient activity schedule.

We illustrate in Figure 5.1 a sample solution for a problem instance defined on a 4×4 grid, where planning horizon is two periods (i.e. $T = 2$), each point must be covered by at least two sensors (i.e. $d_j = 2$), sensing and communication ranges are both equal to one (i.e. $r_k^s = r_k^c = 1$), and the number of sinks is equal to two (i.e. $S = 2$). Note that some of the sensors are in active mode in both periods, whereas some of them are standing by in one of them.

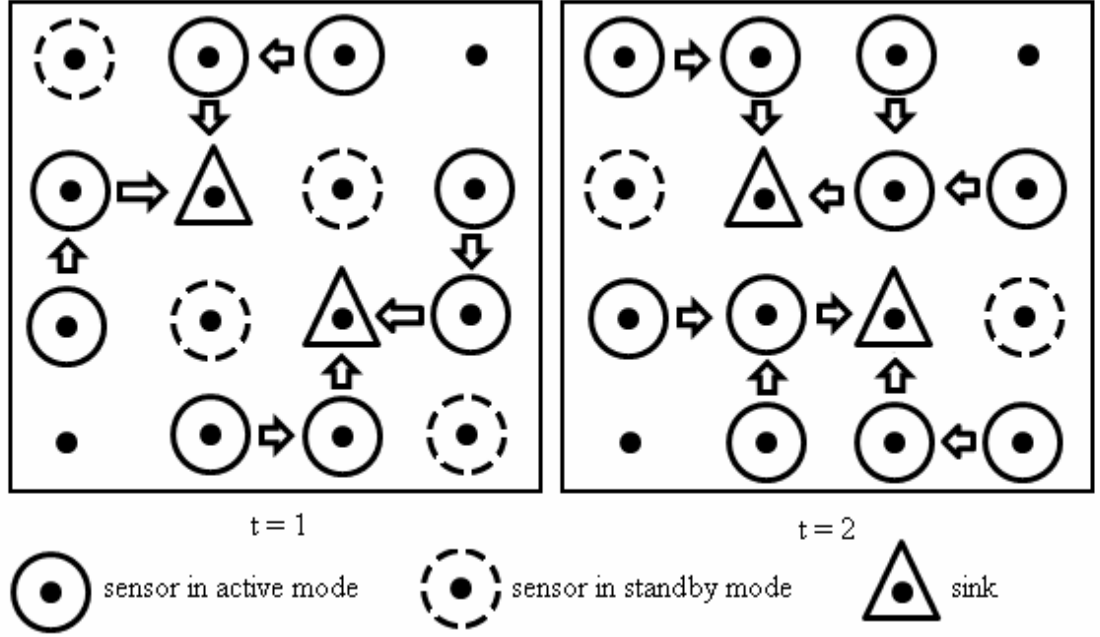


Figure 5.1. A sample sensor network operating for $T = 2$ periods

5.3. Valid Inequalities

In this section we propose valid inequalities for the convex hull of the mixed integer set defined by constraints (5.2)-(5.14) that we call Ω . It is defined as

$$\Omega = \{\mathbf{x}, \mathbf{z}, \mathbf{s}, \mathbf{f}, \mathbf{g}, L, \mathbf{y} : \mathbf{x}, \mathbf{z}, \mathbf{s}, \mathbf{f}, \mathbf{g}, L, \mathbf{y} \text{ satisfy constraints (5.2) - (5.14)}\}$$

Proposition 10.

$$L - \sum_{t=1}^T y_t = 0 \tag{5.15}$$

$$y_{t-1} \geq y_t \quad t = 2, \dots, T \tag{5.16}$$

are valid inequalities for the convex hull of Ω .

Proof. For an $(\mathbf{x}, \mathbf{z}, \mathbf{s}, \mathbf{f}, \mathbf{g}, L, \mathbf{y}) \in \Omega$, $y_t = 1$ if $t \leq L$, $y_t = 0$ otherwise. This implies (5.15), (5.16). Q.E.D.

Next we show that inequalities

$$\sum_{i=1}^J \sum_{k=1}^K E_k x_{ik} \geq E_{lb} L \quad (5.17)$$

and

$$\sum_{i=1}^J \sum_{k=1}^K \sum_{t=1}^T (e_{ik}^s + h_{ik} e_k^c) z_{ikt} \geq E_{lb} L \quad (5.18)$$

that relate L with the location and activation variables $\mathbf{x}$ and $\mathbf{z}$ using the energy information obtained from the solution of a single period coverage problem are also valid inequalities. Here, E_{lb} is defined as a lower bound on the optimal value of the binary integer programming problem (BIPP)

$$\text{Min} \quad \sum_{i=1}^J \sum_{k=1}^K (e_{ik}^s + h_{ik} e_k^c) w_{ik} \quad (5.19)$$

s.t.

$$\sum_{i=1}^J \sum_{k=1}^K a_{ijk} w_{ik} \geq d_j \quad j = 1, \dots, J \quad (5.20)$$

$$w_{ik} \in \{0, 1\} \quad i = 1, \dots, J; k = 1, \dots, K. \quad (5.21)$$

In other words, E_{lb} is a lower bound on the total energy spent for sensing, processing and transmitting the data generated during a period by a set of sensors satisfying coverage constraints (5.3). Hence, if the lifetime is L , at least $E_{lb}L$ units of total energy has to be spent for sensing, processing and transmitting the data generated. The left hand side of (5.17) is the total initial battery energy of the placed sensors. This has to be larger than the lower bound on the total energy consumed for sensing, processing and transmitting the data generated over the lifetime of the network, which is the right hand side of (5.17). The left hand side of (5.18) is the total energy spent for sensing, processing and transmitting the generated data during T periods. Since L is less than T , this has to be larger than the lower bound on the total energy spent during the lifetime of the network for sensing, processing and transmitting the data generated, which is the right hand side of (5.18).

For a formal discussion let $(\mathbf{x}, \mathbf{z}, \mathbf{s}, \mathbf{f}, \mathbf{g}, L, \mathbf{y})$ be a feasible solution in Ω . $y_t = 1$ for $t \leq L$ from constraints (5.4). Hence for $t \leq L$

$$\sum_{i=1}^J \sum_{k=1}^K a_{ijk} z_{ikt} \geq d_j \quad j = 1, \dots, J \quad (5.22)$$

follows from constraints (5.3). Therefore, $\{z_{ikt} : i = 1, \dots, J; k = 1, \dots, K\}$ is feasible for BIPP (5.19) – (5.21) when $t \leq L$. Hence,

$$\sum_{i=1}^J \sum_{k=1}^K (e_{ik}^s + h_{ik} e_k^c) z_{ikt} \geq E_{lb} \quad t = 1, \dots, L \quad (5.23)$$

from which

$$\sum_{t=1}^L \sum_{i=1}^J \sum_{k=1}^K (e_{ik}^s + h_{ik} e_k^c) z_{ikt} \geq E_{lb} L \quad (5.24)$$

follows after summing them up over t . Then, it is possible to change the upper limit of the first summation from L to T due to the nonnegativity of $z_{ikt}, e_{ik}^s, h_{ik}, e_k^c$ and obtain (5.18).

Besides, if $x_{ik} = 0$, then $z_{ikt} = 0$ as a consequence of constraints (5.7). Also, if $x_{ik} = 1$ then $\sum_{t=1}^T (e_{ik}^s + h_{ik} e_k^c) z_{ikt} \leq E_k$ follows from constraints (5.5) and (5.6). These two facts imply

$$\sum_{t=1}^T (e_{ik}^s + h_{ik} e_k^c) z_{ikt} \leq E_k x_{ik} \quad i = 1, \dots, J; k = 1, \dots, K \quad (5.25)$$

which results in

$$\sum_{i=1}^J \sum_{k=1}^K \sum_{t=1}^T (e_{ik}^s + h_{ik} e_k^c) z_{ikt} \leq \sum_{i=1}^J \sum_{k=1}^K E_k x_{ik} \quad (5.26)$$

after summing them up over i and k . Observe that (5.26) and (5.24) imply (5.17). As a consequence of this discussion the following proposition has been shown

Proposition 11. *Inequalities (5.17) and (5.18) are valid for the convex hull of Ω .*

The third proposition introduced in this section simply provides a lower bound on the total cost of active sensors in each period t by using the information obtained from the solution of a single period coverage problem. Let us consider the BIPP

$$\text{Min} \quad \sum_{i=1}^J \sum_{k=1}^K c_{ik} w_{ik} \quad (5.27)$$

s.t.

$$\sum_{i=1}^J \sum_{k=1}^K a_{ijk} w_{ik} \geq d_j \quad j = 1, \dots, J \quad (5.28)$$

$$w_{ik} \in \{0, 1\} \quad i = 1, \dots, J; k = 1, \dots, K \quad (5.29)$$

and let C_{lb} denote a lower bound on its optimal value. Then,

Proposition 12.

$$\sum_{i=1}^J \sum_{k=1}^K c_{ik} z_{ikt} \geq C_{lb} y_t \quad t = 1, \dots, T \quad (5.30)$$

is a valid inequality for the convex hull of Ω .

Proof. If $t \leq L$ then $y_t = 1$ from constraints (5.4). Intuitively, (5.30) formulates the fact that in every period $t \leq L$, the cost of the active sensors has to be larger than the minimum cost of a covering sensor set. For a formal proof consider a vector $(\mathbf{x}, \mathbf{z}, \mathbf{s}, \mathbf{f}, \mathbf{g}, L, \mathbf{y})$ of Ω . If $y_t = 1$, then

$$\sum_{i=1}^J \sum_{k=1}^K a_{ijk} z_{ikt} \geq d_j \quad j = 1, \dots, J \quad (5.31)$$

follows from constraints (5.3). Hence, $\{z_{ikt} : i = 1, \dots, J; k = 1, \dots, K\}$ is a feasible solution of the BIPP (5.27) – (5.29), which implies

$$\sum_{i=1}^J \sum_{k=1}^K c_{ik} z_{ikt} \geq C_{lb} y_t.$$

Notice that this inequality trivially holds for $y_t = 0$ since $c_{ik} \geq 0$ and $z_{ikt} \geq 0$. Q.E.D.

The last proposition describes a valid inequality that sets an upper bound on the total number of active periods of a type- k sensor placed at point i of the sensor field.

Proposition 13.

$$\sum_{t=1}^T z_{ikt} \leq \left\lfloor \frac{E_k}{e_{ik}^s + h_{ik}e_k^c} \right\rfloor x_{ik} \quad i = 1, \dots, J; \quad k = 1, \dots, K \quad (5.32)$$

are valid inequalities for the convex hull of Ω .

Proof. Inequality (5.32) expresses the fact that a type- k sensor cannot be active at point i if it is not placed. $\left\lfloor \frac{E_k}{e_{ik}^s + h_{ik}e_k^c} \right\rfloor$ is the upper bound on the number of periods during which a type- k sensor can satisfy energy requirement for sensing, processing and transmitting the data generated. Hence, inequality (5.32) sets also an upper bound on the number of active periods of sensor (i, k) . Formal proof of validity is as follows. Given a vector $(\mathbf{x}, \mathbf{z}, \mathbf{s}, \mathbf{f}, \mathbf{g}, L, \mathbf{y})$ in Ω , if $x_{ik} = 1$, then

$$\sum_{t=1}^T (e_{ik}^s + h_{ik}e_k^c) z_{ikt} \leq E_k \quad (5.33)$$

can be obtained as a consequence of constraints (5.5) and (5.6). In addition, the division by $e_{ik}^s + h_{ik}e_k^c$ of both sides of the last inequality gives

$$\sum_{t=1}^T z_{ikt} \leq \frac{E_k}{e_{ik}^s + h_{ik}e_k^c} \quad (5.34)$$

from which

$$\sum_{t=1}^T z_{ikt} \leq \left\lfloor \frac{E_k}{e_{ik}^s + h_{ik}e_k^c} \right\rfloor$$

can be written, since any $\mathbf{z}$ satisfying the constraints of Ω is integer. Finally, if $x_{ik} = 0$, then $z_{ikt} = 0$ for all t as a consequence of constraints (5.7).

6. A BRANCH AND PRICE ALGORITHM

6.1. Introduction

In this section we propose an exact solution procedure for solving LSRP. We first reformulate the problem. Then, we develop an efficient solution algorithm in order to solve the LP relaxation of the reformulation. Finally, we devise a branch and price algorithm that computes an optimal solution of LSRP.

6.2. Reformulation

LSRP can be equivalently reformulated in the following slightly different way as LSRP':

$$\max \quad \sum_{t=1}^T y_t \quad (6.1)$$

s.t.

$$\sum_{t=1}^T \left(e_{ik}^s z_{ikt} + e_k^r \sum_{\substack{j=1 \\ j \neq i}}^J \sum_{l=1}^K b_{jil} f_{jlik} + \sum_{\substack{j=1 \\ j \neq i}}^J e_k^c \sum_{l=1}^K b_{ijk} f_{ikjlt} + \sum_{j=1}^J e_k^c b_{ijk} g_{ikjt} \right) \leq E_k x_{ik} \quad \text{for all } i, k \quad (6.2)$$

$$(5.3), (5.5), (5.8) - (5.14), (5.30).$$

The objective function (6.1) expresses network lifetime as the sum of the time periods within the lifetime. Expressing lifetime in this way makes constraints (5.4) redundant. Notice that when x_{ik} is zero, z_{ikt} is zero due to constraints (6.2). Therefore, constraints (5.7) are not needed. Constraints (5.30) are included in order to strengthen the formulation.

When constraints (5.13) and (6.2) are relaxed with Lagrangean multipliers θ and α , and scaled so that the largest coefficient becomes one, the following Lagrangean

problem (LG) is obtained:

LG:

$$\begin{aligned} \max \quad & \sum_{t=1}^T y_t + \sum_{i=1}^J \sum_{k=1}^K \alpha_{ik} \left(x_{ik} - \sum_{t=1}^T \frac{\gamma_{ikt}}{E_k} \right) + \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{t=1}^T \theta_{ikjt} \left(s_j - \frac{1}{v_k} g_{ikjt} \right) \\ \text{s.t.} \quad & (5.3), (5.5), (5.8) - (5.12), (5.14), (5.30). \end{aligned} \quad (6.3)$$

In (6.3), $\gamma_{ikt} = e_{ik}^s z_{ikt} + e_k^r \sum_{\substack{j=1 \\ j \neq i}}^J \sum_{l=1}^K b_{jil} f_{jlkt} + \sum_{\substack{j=1 \\ j \neq i}}^J e_k^c \sum_{l=1}^K b_{ijk} f_{ikjlt} + \sum_{j=1}^J e_k^c b_{ijk} g_{ikjt}$, which is the expression in the summation term of Constraints (6.2), formulating the total energy spent in period t by sensor (i, k) . We denote by $LG(\boldsymbol{\alpha}, \boldsymbol{\theta})$ the optimal value of problem LG for $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ given.

LG can be decomposed into $T + 2$ subproblems. The first T for each period t are

LG_t^1 :

$$\max \quad y_t - \sum_{i=1}^J \sum_{k=1}^K \alpha_{ik} \frac{\gamma_{ikt}}{E_k} - \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \frac{\theta_{ikjt}}{v_k} g_{ikjt} \quad (6.4)$$

$$\text{s.t.} \quad (5.3), (5.5), (5.10) - (5.12), (5.30) \text{ expressed for } t \text{ only.} \quad (6.5)$$

$$z_{ikt}, y_t \in \{0, 1\}, f_{ikjlt} \geq 0, g_{ikjt} \geq 0. \quad (6.6)$$

We denote by $LG_t^1(\boldsymbol{\alpha}, \boldsymbol{\theta})$ the optimal value of problem LG_t^1 for given $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$. The second subproblem is

LG₂ :

$$\max \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{t=1}^T \theta_{ikjt} s_j \quad (6.7)$$

$$b_{ijk}=1$$

$$\text{s.t. (5.9)} \quad (6.8)$$

$$s_i \in \{0, 1\}. \quad (6.9)$$

We denote by $LG_2(\boldsymbol{\theta})$ the optimal value of problem LG₂ with Lagrangean multiplier $\boldsymbol{\theta}$. Finally, the third subproblem is

LG₃ :

$$\max \sum_{i=1}^J \sum_{k=1}^K \alpha_{ik} x_{ik} \quad (6.10)$$

$$\text{s.t. (5.8)} \quad (6.11)$$

$$x_{ik} \in \{0, 1\}. \quad (6.12)$$

We denote by $LG_3(\boldsymbol{\alpha})$ the optimal value of problem LG₃ when $\boldsymbol{\alpha}$ is fixed. One can obtain the optimal value of LG by simply summing the optimal objective values of the subproblems as

$$LG(\boldsymbol{\alpha}, \boldsymbol{\theta}) = \sum_{t=1}^T LG_t^1(\boldsymbol{\alpha}, \boldsymbol{\theta}) + LG_2(\boldsymbol{\theta}) + LG_3(\boldsymbol{\alpha}). \quad (6.13)$$

Since the number of integer variables increases with T in the original formulation of LSRP, this choice of relaxation strategy reduces considerably this negative effect of increasing T on the efficiency.

Let $(\mathbf{z}^{h^{1t}}, \mathbf{y}^{h^{1t}}, \mathbf{f}^{h^{1t}}, \mathbf{g}^{h^{1t}}) \mid h^{1t} \in H^{1t}, t \in [1, 2, \dots, T]$ be the set of extreme points of the convex hull of the mixed-integer set defined by (6.5) and (6.6); $\mathbf{s}^{h^2} \mid h^2 \in H^2$ be the set of extreme points of the convex hull of the mixed-integer set defined by (6.8) and (6.9); $\mathbf{x}^{h^3} \mid h^3 \in H^3$ be the set of extreme points of the convex hull of the mixed-integer set defined by (6.11) and (6.12).

The so called Lagrangean dual problem (LD) of minimizing $LG(\boldsymbol{\alpha}, \boldsymbol{\theta})$ over $\boldsymbol{\alpha} \geq \mathbf{0}$, $\boldsymbol{\theta} \geq \mathbf{0}$ can now be written as

LD:

$$\min \left(\sum_{t=1}^T \beta_t^1 \right) + \beta^2 + \beta^3 \quad (6.14)$$

s.t.

$$y_t^{h^{1t}} - \sum_{i=1}^J \sum_{k=1}^K \alpha_{ik} \frac{\gamma_{ikt}^{h^{1t}}}{E_k} - \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \frac{\theta_{ikjt}}{v_k} g_{ikjt}^{h^{1t}} \leq \beta_t^1 \quad h^{1t} \in H^{1t}, t \in [1, \dots, T] \quad (6.15)$$

$$\sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{t=1}^T \theta_{ikjt} s_j^{h^2} \leq \beta^2 \quad h^2 \in H^2 \quad (6.16)$$

$$\sum_{i=1}^J \sum_{k=1}^K \alpha_{ik} x_{ik}^{h^3} \leq \beta^3 \quad h^3 \in H^3 \quad (6.17)$$

$$\alpha_{ik} \geq 0, \theta_{ikjt} \geq 0 \quad b_{ijk} = 1 \quad (6.18)$$

$$\beta_t^1, \beta^2, \beta^3 \text{ unrestricted.} \quad (6.19)$$

We denote by LD^* the optimal value of problem LD. When we take the dual of LD, we obtain the LP relaxation of the reformulation of LSRP that we call RLSRP.

RLSRP:

$$\max \quad \sum_{t=1}^T \sum_{h^{1t} \in H^{1t}} y_t^{h^{1t}} d^{h^{1t}} \quad (6.20)$$

s.t.

$$\sum_{t=1}^T \sum_{h^{1t} \in H^{1t}} \frac{\gamma_{ikt}^{h^{1t}}}{E_k} d^{h^{1t}} - \sum_{h^3 \in H^3} x_{ik}^{h^3} d^{h^3} \leq 0 \quad i, k \quad (6.21)$$

$$\sum_{h^{1t} \in H^{1t}} \frac{g_{ikjt}^{h^{1t}}}{v_k} d^{h^{1t}} - \sum_{h^2 \in H^2} s_j^{h^2} d^{h^2} \leq 0 \quad i, k, j, t \text{ with } b_{ijk} = 1 \quad (6.22)$$

$$\sum_{h^{1t} \in H^{1t}} d^{h^{1t}} = 1 \quad t \quad (6.23)$$

$$\sum_{h^2 \in H^2} d^{h^2} = 1 \quad (6.24)$$

$$\sum_{h^3 \in H^3} d^{h^3} = 1 \quad (6.25)$$

$$d^{h^{1t}}, d^{h^2}, d^{h^3} \geq 0. \quad (6.26)$$

The reformulation of LSRP contains integrality constraints $\sum_{h^{1t} \in H^{1t}} d^{h^{1t}} z_{ikt}^{h^{1t}} \in \{0, 1\}$ for all i, k, t ; $\sum_{h^2 \in H^2} s_j^{h^2} d^{h^2} \in \{0, 1\}$ for all j ; $\sum_{h^3 \in H^3} x_{ik}^{h^3} d^{h^3} \in \{0, 1\}$ for all i, k ; $\sum_{h^{1t} \in H^{1t}} d^{h^{1t}} y_t^{h^{1t}} \in \{0, 1\}$ for all t . If $z_{ikt}^f = \sum_{h^{1t} \in H^{1t}} d_*^{h^{1t}} z_{ikt}^{h^{1t}} \in \{0, 1\}$ for all i, k, t ; $s_j^f = \sum_{h^2 \in H^2} s_j^{h^2} d_*^{h^2} \in \{0, 1\}$ for all j ; $x_{ik}^f = \sum_{h^3 \in H^3} x_{ik}^{h^3} d_*^{h^3} \in \{0, 1\}$ for all i, k ; $y_t^f = \sum_{h^{1t} \in H^{1t}} d_*^{h^{1t}} y_t^{h^{1t}} \in \{0, 1\}$ for all t then,

$$\left(\mathbf{x}^f, \mathbf{z}^f, \mathbf{s}^f, \mathbf{y}^f, f_{ikjlt}^f = \sum_{h^{1t} \in H^{1t}} d_*^{h^{1t}} f_{ikjlt}^{h^{1t}}, g_{ikjt}^f = \sum_{h^{1t} \in H^{1t}} d_*^{h^{1t}} g_{ikjt}^{h^{1t}} \right) \quad (6.27)$$

is an optimal solution of the reformulation LSRP'. Here, $d_*^{h^{1t}}, d_*^{h^2}, d_*^{h^3}$ is an optimal solution of RLSRP.

6.3. Solution Procedure

6.3.1. Solution of the Reformulation

In this section a solution procedure for RLSRP is presented. It combines subgradient optimization and the weighted Dantzig-Wolfe decomposition algorithm of Wentges

[64] as done in Klose and Görtz [65]. In its first step called initial subgradient phase, a subgradient algorithm that is based on the deflected and the conditional subgradient algorithms as described in Guta [66] is applied in order to solve LD². The subgradient algorithm is summarized in Figure 6.1, where α^n, θ^n denote the values of Lagrange multiplier vectors α and θ at iteration n , $(z^n, y^n, f^n, g^n, s^n, x^n)$ indicate the solution of LG at α^n, θ^n . $\widehat{LG}$ stores the lowest $LG(\alpha^n, \theta^n)$ value obtained as the algorithm progress. $\widehat{\alpha}, \widehat{\theta}$ are the Lagrange multiplier values giving $\widehat{LG}$, i.e. $LG(\widehat{\alpha}, \widehat{\theta}) = \widehat{LG}$. In the initial subgradient phase, at iteration n , first the subproblems LG₁^t, LG₂, and LG₃ are solved. If any of these subproblems is infeasible or $LG(\alpha^n, \theta^n)$ is less than one, LSRP' is infeasible or has an optimal solution with zero lifetime value. If $LG(\alpha^n, \theta^n)$ is less than $\widehat{LG}$ then, $\widehat{\alpha}, \widehat{\theta}$ and $\widehat{LG}$ are updated. If $\widehat{LG}$ does not change in m consecutive iterations step size parameter μ^n is halved. As mentioned in Figure 6.1, at each iteration of the subgradient algorithm, a feasible solution is constructed from the solution of LG at α^n, θ^n that does not necessarily satisfy constraints (5.13) and (6.2). The algorithm called *Dual Heuristic* (DH) is designed for this purpose. It is summarized in Figure 6.5. In its first step, $(z^n, y^n, f^n, g^n, s^n, x^n)$ is assigned to vector $(\bar{z}, \bar{y}, \bar{f}, \bar{g}, \bar{s}, \bar{x})$ storing the constructed feasible solution. In the second step, $\bar{y}_t$ is set to 0 for periods t in which there exists an active sensor that is not located. In the third step, if there is no sink at point j , sensor-to-sink flow value $\bar{g}_{ikjt}$ from sensor (i, k) to point j is routed to the sinks at other points within the transmission range of the sensor (i, k) without exceeding flow capacity v_k . Finally, in the fourth step, sensors violating energy constraint (6.2) are found. In Figure 6.5, σ_t denotes the number of sensors that are active in period t and that violate energy constraint (6.2). Among all period t with $\bar{y}_t = 1$, the one with maximum σ_t value, t^* , is found and $\bar{y}_{t^*}$ is set to 0. This is repeated until all sensors located satisfy energy constraint (6.2). The feasible solution constructed using DH at iteration n of the initial subgradient phase is shown by $(\bar{z}^n, \bar{y}^n, \bar{f}^n, \bar{g}^n, \bar{s}^n, \bar{x}^n)$. $\overline{LB}^n$ is the corresponding network lifetime value. $\widehat{LB}$ stores the highest network lifetime value found. Assigning lifetime value of good feasible solution of LSRP to LB^* at the beginning of the phase accelerates convergence of the subgradient algorithm. Therefore, network lifetime value given by a heuristic called Disjoint Set Heuristic (DSH) denoted as L_{DSH} is assigned to $\widehat{LB}$. The heuristic

²A minor flaw in the convergence proof presented in Guta [66] which we correct in Appendix B

finds connected disjoint sensor sets that satisfy the coverage requirement and that have minimum cost in its first two steps. In the feasible solution constructed from them, at any time period, only one sensor set is active, i.e. sensors that are element of the set are in active mode, other sensors are all in standby mode. The active set is replaced by another set at the end of the time period when one sensor in the active set runs out of energy. In the active periods of a set, sensor-to-sink flow quantities are found by solving a LP maximizing the set's lifetime in the third step of DSH. Final step removes cycles from these flow quantities. DSH will be explained in detail in Chapter 9. In order to find Lagrange multipliers at iteration $n + 1$, $\alpha^{n+1}, \theta^{n+1}$, first subgradient vector (ϵ^n, ζ^n) is calculated. Then the conditional subgradient vector $(\bar{\epsilon}^n, \bar{\zeta}^n)$ is determined. The next step is the calculation of the deflection coefficient at iteration n , η^n . Direction vector at iteration n $(\tilde{\epsilon}^n, \tilde{\zeta}^n)$ is the conditional gradient vector if the deflected direction is an infeasible direction. It is the deflected direction otherwise.

Even if optimal solution of LD is obtained in the initial subgradient phase, subsets of H^{1t}, H^2, H^3 giving that solution have to be formed in order to solve optimally RLSRP. For this purpose, weighted Dantzig-Wolfe decomposition algorithm of Wentges [64] is employed. This constitutes the second phase of Algorithm in Figure 6.1, at each iteration $\tilde{n}$ of which, problem LD is solved with a subset of H^{1t}, H^2, H^3 . That restricted LD is denoted by $LD_r^{\tilde{n}}$ in Figure 6.1. Notice that constraint $(\sum_{t=1}^T \beta_t^1) + \beta^2 + \beta^3 \geq 0$ is added before solving $LD_r^{\tilde{n}}$ to its constraint set in order to avoid unboundedness. The addition of this constraint is validated by the fact that $LD \geq 1$ if LSRP' has a feasible solution with positive lifetime value. Notice also that problem LG is solved at

$$\alpha^{\tilde{n}^\lambda} = \lambda \alpha^{\tilde{n}} + (1 - \lambda) \hat{\alpha} \quad (6.28)$$

$$\theta^{\tilde{n}^\lambda} = \lambda \theta^{\tilde{n}} + (1 - \lambda) \hat{\theta} \quad (6.29)$$

for a $\lambda \in (0, 1]$ rather than at $\alpha^{\tilde{n}}, \theta^{\tilde{n}}$ as suggested by Wentges [64]. Moreover, if $(z^{\tilde{n}^\lambda}, y^{\tilde{n}^\lambda}, f^{\tilde{n}^\lambda}, g^{\tilde{n}^\lambda}, s^{\tilde{n}^\lambda}, x^{\tilde{n}^\lambda})$ obtained is feasible for $LD_r^{\tilde{n}}$ at $(\alpha^{\tilde{n}}, \theta^{\tilde{n}}, \beta_t^{1\tilde{n}}, \beta^{2\tilde{n}}, \beta^{3\tilde{n}})$ then λ is increased by $\Delta\lambda$ and LG is solved at $\alpha^{\tilde{n}^\lambda}, \theta^{\tilde{n}^\lambda}$ again. This is based on the following proposition stating that $\widehat{LG}$ is decreased at the previous $\alpha^{\tilde{n}^\lambda}, \theta^{\tilde{n}^\lambda}$ multiplier values.

Proposition 14. If $\left(\mathbf{z}^{\tilde{n}^\lambda}, \mathbf{y}^{\tilde{n}^\lambda}, \mathbf{f}^{\tilde{n}^\lambda}, \mathbf{g}^{\tilde{n}^\lambda}, \mathbf{s}^{\tilde{n}^\lambda}, \mathbf{x}^{\tilde{n}^\lambda}\right)$ obtained is feasible for $LD_r^{\tilde{n}}$ at $(\boldsymbol{\alpha}^{\tilde{n}}, \boldsymbol{\theta}^{\tilde{n}}, \beta_t^{1\tilde{n}}, \beta^{2\tilde{n}}, \beta^{3\tilde{n}})$ then $LG(\boldsymbol{\alpha}^{\tilde{n}^\lambda}, \boldsymbol{\theta}^{\tilde{n}^\lambda}) \leq \widehat{LG} - \lambda(\widehat{LG} - LD_r^{\tilde{n}})$.

Proof. Clearly,

$$\begin{aligned}
LG(\boldsymbol{\alpha}^{\tilde{n}^\lambda}, \boldsymbol{\theta}^{\tilde{n}^\lambda}) &= \lambda \left\{ \sum_{t=1}^T \left(y_t^{\tilde{n}^\lambda} - \sum_{i=1}^J \sum_{k=1}^K \alpha_{ik}^{\tilde{n}} \gamma_{ikt}^{\tilde{n}^\lambda} / E_k \right. \right. \\
&\quad \left. \left. - \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J b_{ijk} \theta_{ikjt}^{\tilde{n}} g_{ikjt}^{\tilde{n}^\lambda} / v_k \right) + \right. \\
&\quad \left. \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{t=1}^T b_{ijk} \theta_{ikjt}^{\tilde{n}} s_j^{\tilde{n}^\lambda} + \sum_{i=1}^J \sum_{k=1}^K \alpha_{ik}^{\tilde{n}} x_{ik}^{\tilde{n}^\lambda} \right\} + \\
&\quad (1 - \lambda) \left\{ \sum_{t=1}^T \left(y_t^{\tilde{n}^\lambda} - \sum_{i=1}^J \sum_{k=1}^K \alpha_{ik}^* \gamma_{ikt}^{\tilde{n}^\lambda} / E_k \right. \right. \\
&\quad \left. \left. - \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J b_{ijk} \theta_{ikjt}^* g_{ikjt}^{\tilde{n}^\lambda} / v_k \right) \right. \\
&\quad \left. + \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{t=1}^T b_{ijk} \theta_{ikjt}^* s_j^{\tilde{n}^\lambda} + \sum_{i=1}^J \sum_{k=1}^K \alpha_{ik}^* x_{ik}^{\tilde{n}^\lambda} \right\} \quad (6.30)
\end{aligned}$$

by definition of $\boldsymbol{\alpha}^{\tilde{n}^\lambda}, \boldsymbol{\theta}^{\tilde{n}^\lambda}$. This expression is less than $\lambda LD_r^{\tilde{n}} + (1 - \lambda) \widehat{LG}$ because $\left(\mathbf{z}^{\tilde{n}^\lambda}, \mathbf{y}^{\tilde{n}^\lambda}, \mathbf{f}^{\tilde{n}^\lambda}, \mathbf{g}^{\tilde{n}^\lambda}, \mathbf{s}^{\tilde{n}^\lambda}, \mathbf{x}^{\tilde{n}^\lambda}\right)$ is feasible for $LD_r^{\tilde{n}}$ at $(\boldsymbol{\alpha}^{\tilde{n}}, \boldsymbol{\theta}^{\tilde{n}}, \beta_t^{1\tilde{n}}, \beta^{2\tilde{n}}, \beta^{3\tilde{n}})$ and $\left(\mathbf{z}^{\tilde{n}^\lambda}, \mathbf{y}^{\tilde{n}^\lambda}, \mathbf{f}^{\tilde{n}^\lambda}, \mathbf{g}^{\tilde{n}^\lambda}, \mathbf{s}^{\tilde{n}^\lambda}, \mathbf{x}^{\tilde{n}^\lambda}\right)$ is a feasible point of problem LG with $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\theta}}$. Q.E.D.

Proposition 14 is very similar to Lemma 3.1 of Wentges [64]. The difference is that in Wentges [64] the condition of the proposition is given as the identity of the constraints related to h^{1t}, h^2, h^3 corresponding to $\left(\mathbf{z}^{\tilde{n}^\lambda}, \mathbf{y}^{\tilde{n}^\lambda}, \mathbf{f}^{\tilde{n}^\lambda}, \mathbf{g}^{\tilde{n}^\lambda}, \mathbf{s}^{\tilde{n}^\lambda}, \mathbf{x}^{\tilde{n}^\lambda}\right)$ to those of $LD_r^{\tilde{n}}$. If $\left(\mathbf{z}^{\tilde{n}^\lambda}, \mathbf{y}^{\tilde{n}^\lambda}, \mathbf{f}^{\tilde{n}^\lambda}, \mathbf{g}^{\tilde{n}^\lambda}, \mathbf{s}^{\tilde{n}^\lambda}, \mathbf{x}^{\tilde{n}^\lambda}\right)$ obtained is not feasible for $LD_r^{\tilde{n}}$ at $(\boldsymbol{\alpha}^{\tilde{n}}, \boldsymbol{\theta}^{\tilde{n}}, \beta_t^{1\tilde{n}}, \beta^{2\tilde{n}}, \beta^{3\tilde{n}})$ then h^{1t}, h^2, h^3 corresponding to $\left(\mathbf{z}^{\tilde{n}^\lambda}, \mathbf{y}^{\tilde{n}^\lambda}, \mathbf{f}^{\tilde{n}^\lambda}, \mathbf{g}^{\tilde{n}^\lambda}, \mathbf{s}^{\tilde{n}^\lambda}, \mathbf{x}^{\tilde{n}^\lambda}\right)$ are included in $LD_r^{\tilde{n}}$. Additional subgradient steps are performed as suggested by Klose and Görtz [65] starting with $\boldsymbol{\alpha}^1 = \boldsymbol{\alpha}^{\tilde{n}^\lambda}, \boldsymbol{\theta}^1 = \boldsymbol{\theta}^{\tilde{n}^\lambda}$ and $\mu^1 = 2.0$.

Weighted Dantzig-Wolfe decomposition phase terminates if $|\widehat{LG} - \widehat{LB}| < \varepsilon$ or $|\widehat{LG} - LD_r^{\tilde{n}}| < \varepsilon$. As indicated in the final phase of the algorithm in Figure 6.1, in the former case an optimal solution of LSRP' is found in step 3.1. In the latter case an optimal solution of RLSRP can be found by solving RLSRP with $h^{1t} \in H_n^{1t}, h^2 \in H_n^2, h^3 \in H_n^3$ of $LD_r^{\tilde{n}}$ after adding h^{1t}, h^2, h^3 corresponding to the solution of LG_t^1 at $\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\theta}}$ for all t ; LG_2 at $\hat{\boldsymbol{\theta}}$ and LG_3 at $\hat{\boldsymbol{\alpha}}$ in step 3.2.1. Here, H_n^{1t}, H_n^2, H_n^3 denote the subset

of H^{1t}, H^2, H^3 used in $LD_r^{\tilde{n}}$ respectively. A feasible solution to LSRP' is also constructed from an optimal solution to RLSRP by employing an algorithm called *Primal Heuristic* (PH). It is summarized in Figure 6.6 where an optimal solution to RLSRP corresponding to $h^{1t} \in H_n^{1t}, h^2 \in H_n^2, h^3 \in H_n^3$ after adding h^{1t}, h^2, h^3 corresponding to the solution of LG_t^1 at $\hat{\alpha}, \hat{\theta}$ for all t ; LG_2 at $\hat{\theta}$ and LG_3 at $\hat{\alpha}$ is indicated by $d_*^{h^{1t}}, d_*^{h^2}, d_*^{h^3}$. It is very important to notice that in Step 2.2.1 of the weighted Dantzig-Wolfe decomposition phase, $LD_r^{\tilde{n}}$ is solved after adding the constraint $\left(\sum_{t=1}^T \beta_t^1 + \beta^2 + \beta^3 \geq 0\right)$. This is based on the assumption that LSRP' has a feasible solution with positive life-time value, and on the fact that $LD \geq 0$ in that case. Consequently, if LSRP' has a feasible solution, then, an optimal solution to RLSRP can be found by solving RLSRP with $h^{1t} \in H_n^{1t}, h^2 \in H_n^2, h^3 \in H_n^3$ of $LD_r^{\tilde{n}}$ after adding h^{1t}, h^2, h^3 corresponding to the solution of LG_t^1 at $\hat{\alpha}, \hat{\theta}$ for all t ; LG_2 at $\hat{\theta}$ and LG_3 at $\hat{\alpha}$. In fact, this problem is the dual of $LD_r^{\tilde{n}}$ enlarged with the additional h^{1t}, h^2, h^3 corresponding to the solution of LG_t^1 at $\hat{\alpha}, \hat{\theta}$ for all t ; LG_2 at $\hat{\theta}$ and LG_3 at $\hat{\alpha}$. Its solution $d_*^{h^{1t}}, d_*^{h^2}, d_*^{h^3}$, when enlarged such that $d^{h^{1t}} = 0, d^{h^2} = 0, d^{h^3} = 0$ for h^{1t}, h^2, h^3 that were omitted, is a feasible solution to RLSRP. Objective value corresponding to this feasible solution is equal to $\widehat{LG}$. Therefore, it is an optimal solution to RLSRP. Because in each iteration $\tilde{n}$ of the while loop in Step 2.2, either a new element of H^{1t} or H^2 or H^3 is found or the gap $\widehat{LG} - LD_r^{\tilde{n}}$ reduces $100 \times \lambda$ percent, SR terminates in a finite number of iterations.

6.3.2. A Branch and Price Algorithm

When $\mathbf{x}^f, \mathbf{s}^f, \mathbf{z}^f, \mathbf{y}^f$ described in (6.27) are not integer, a branch and price algorithm (Barnhart et al. [67]) is used to obtain an optimal solution to LSRP. It is summarized in Figure 6.7, where L_{LB} denotes the highest feasible objective value found, and L_{UB} indicates the smallest upper bound value for the optimal objective of LSRP. In the first step of the algorithm in Figure 6.7, LSRP' is solved using the algorithm in Figure 6.1. This first problem without any integrality restrictions on any component of the vectors $\mathbf{x}^f, \mathbf{s}^f, \mathbf{z}^f, \mathbf{y}^f$ constitutes the initial node of the brach-and-price tree. It is the first element of *node list* being the set containing nodes. Therefore, in the second step of the algorithm in Figure 6.7, $\widehat{LB}$ is assigned to L_{LB} , $\widehat{LG}$ is assigned to L_{UB} , and *node list* is $\{0\}$. Also, ω_0 , designating the optimal value of the problem associated

1.Initial subgradient phase:

- 1.1. Set $\alpha_{ik}^1 = 0, \bar{\epsilon}_{ik}^0 = 0$ for all $i, k; \theta_{ikjt}^1 = 0, \bar{\zeta}_{ikjt}^0 = 0$ for all i, k, j, t s.t. $b_{ijk} = 1$;
 $\widehat{LB} = L_{DSH}; \widehat{LG} = \infty; \mu^1 = 2.0, n = 1$
 - 1.2. While ($n \leq N$) Do
 - 1.2.1. Solve LG_t^1 with α^n, θ^n for all t ; LG_2 with θ^n and LG_3 with α^n to obtain
 $(\mathbf{z}^n, \mathbf{y}^n, \mathbf{f}^n, \mathbf{g}^n, \mathbf{s}^n, \mathbf{x}^n)$ and $LG(\alpha^n, \theta^n)$
 - 1.2.2. If (any of these problems is infeasible or $\widehat{LG} < 1$) Then LSRP'
 is infeasible or has zero as optimal objective stop
 - 1.2.3. If ($LG(\alpha^n, \theta^n) < \widehat{LG}$) Then $\hat{\alpha} = \alpha^n, \hat{\theta} = \theta^n$, and $\widehat{LG} = LG(\alpha^n, \theta^n)$
 - 1.2.3.1 If $\widehat{LG}$ does not change in m consecutive iterations Then $\mu^n \leftarrow \mu^n/2$
 - 1.2.4. Construct a feasible solution $(\bar{\mathbf{z}}^n, \bar{\mathbf{y}}^n, \bar{\mathbf{f}}^n, \bar{\mathbf{g}}^n, \bar{\mathbf{s}}^n, \bar{\mathbf{x}}^n)$
 from $(\mathbf{z}^n, \mathbf{y}^n, \mathbf{f}^n, \mathbf{g}^n, \mathbf{s}^n, \mathbf{x}^n)$ using DH
 - 1.2.5. If ($\widehat{LB}^n > \widehat{LB}$) Then set $(\mathbf{z}^*, \mathbf{y}^*, \mathbf{f}^*, \mathbf{g}^*, \mathbf{s}^*, \mathbf{x}^*) = (\bar{\mathbf{z}}^n, \bar{\mathbf{y}}^n, \bar{\mathbf{f}}^n, \bar{\mathbf{g}}^n, \bar{\mathbf{s}}^n, \bar{\mathbf{x}}^n)$
 and $\widehat{LB} = \widehat{LB}^n$
 - 1.2.6. If ($|\widehat{LG} - \widehat{LB}| < \varepsilon$) Then Go to 3.1
 - 1.2.7.1. Calculate $\epsilon_{ik}^n = x_{ik}^n - \sum_{t=1}^T \frac{\gamma_{ikt}^n}{E_k}$ for all i, k and $\zeta_{ikjt}^n = s_j^n - \frac{g_{ikjt}^n}{v_k}$ for all
 i, k, j, t , and $b_{ijk} = 1$
 - 1.2.7.2. Set $\bar{\epsilon}_{ik}^n = \begin{cases} 0 & \text{if } \alpha_{ik}^n \text{ is 0 and } \epsilon_{ik}^n > 0 \\ \epsilon_{ik}^n & \text{otherwise} \end{cases}$
 and $\bar{\zeta}_{ikjt}^n = \begin{cases} 0 & \text{if } \theta_{ikjt}^n \text{ is 0 and } \zeta_{ikjt}^n > 0 \\ \zeta_{ikjt}^n & \text{otherwise} \end{cases}$
 - 1.2.7.3. Set $\eta^n = \begin{cases} -\frac{1.5(\bar{\epsilon}^n, \bar{\zeta}^n)^T(\bar{\epsilon}^{n-1}, \bar{\zeta}^{n-1})}{\|(\bar{\epsilon}^{n-1}, \bar{\zeta}^{n-1})\|^2} & \text{if } (\bar{\epsilon}^n, \bar{\zeta}^n)^T(\bar{\epsilon}^{n-1}, \bar{\zeta}^{n-1}) < 0 \\ 0 & \text{otherwise} \end{cases}$
-

Figure 6.1. Solution of RLSRP (first part)

with the initial node 0, is given value $\widehat{LG}$, which is the optimal value of RLSRP. As long as *node list* is not empty, one of its elements is selected. Best first node selection strategy is used, i.e. among the nodes i in *node list*, the one with the highest w_i value is

$$\begin{aligned}
1.2.7.4. \quad \text{Set } (\tilde{\epsilon}^n, \tilde{\zeta}^n) &= \begin{cases} (\bar{\epsilon}^n, \bar{\zeta}^n) & \begin{array}{l} \text{if there is } i, k \text{ such that} \\ \alpha_{ik}^n = 0 \text{ and } \bar{\epsilon}_{ik}^n + \eta^n \tilde{\epsilon}_{ik}^{n-1} > 0 \\ \text{or } i, k, j, t \text{ with} \\ \theta_{ikjt}^n = 0 \text{ and } \bar{\zeta}_{ikjt}^n + \eta^n \tilde{\zeta}_{ikjt}^{n-1} > 0 \end{array} \\ (\bar{\epsilon}^n, \bar{\zeta}^n) + \eta^n (\tilde{\epsilon}^{n-1}, \tilde{\zeta}^{n-1}) & \text{otherwise} \end{cases} \\
1.2.7.5. \quad \text{Set } \alpha_{ik}^{n+1} &= \begin{cases} \max \left(\alpha_{ik}^n - \frac{\mu^n (LG(\boldsymbol{\alpha}^n, \boldsymbol{\theta}^n) - \widehat{LB})}{\|(\tilde{\epsilon}^n, \tilde{\zeta}^n)\|^2} \tilde{\epsilon}_{ik}^n, 0 \right) & \text{if } \|(\tilde{\epsilon}^n, \tilde{\zeta}^n)\|^2 > 0 \\ \alpha_{ik}^n & \text{otherwise} \end{cases} \\
\text{Set } \theta_{ikjt}^{n+1} &= \begin{cases} \max \left(\theta_{ikjt}^n - \frac{\mu^n (LG(\boldsymbol{\alpha}^n, \boldsymbol{\theta}^n) - \widehat{LB})}{\|(\tilde{\epsilon}^n, \tilde{\zeta}^n)\|^2} \tilde{\zeta}_{ikjt}^n, 0 \right) & \text{if } \|(\tilde{\epsilon}^n, \tilde{\zeta}^n)\|^2 > 0 \\ \theta_{ikjt}^n & \text{otherwise} \end{cases} \\
1.2.8. \quad n &\leftarrow n + 1 \\
&\text{End While}
\end{aligned}$$

Figure 6.2. Solution of RLSRP (second part)

chosen for further branching, where w_i is the optimal value of RLSRP with additional equality constraints on some components of $\mathbf{x}^f, \mathbf{s}^f, \mathbf{z}^f, \mathbf{y}^f$ as a result of branching that defines the node. Notice that RLSRP with equality constraints on some components of $\mathbf{x}^f, \mathbf{s}^f, \mathbf{z}^f, \mathbf{y}^f$ can be solved with the algorithm in Figure 6.1 after minor modifications. First, those equality constraints should be included in the subproblems that contain the variable involved. For instance, an equality constraint, $x_{ik} = 1$, should be included in the third subproblem LG_3 , if it is in the constraint set of the node that results from branching. Moreover, in applying algorithms DH and PH, these equality constraints should be taken into account. For instance, if $y_t = 1$ is a constraint of the node then in Steps 2, 3 and 4 of DH, and in steps 3 and 4 of PH y_t cannot be set to zero. Also, DSH algorithm used in the initial subgradient phase should be modified so that it is in accordance with the additional equality constraints. Apart from these slight modifications, the basic structure of the algorithm in Figure 6.1 and Proposition 14 forming its foundation are not altered by the additional equality constraints involving

2. Weighted Dantzig-Wolfe decomposition phase:

- 2.1. Set $\tilde{n} = 1$, initialize LD_r^1 with h^{1t}, h^2, h^3 corresponding to the elements of $\{(\mathbf{z}^n, \mathbf{y}^n, \mathbf{f}^n, \mathbf{g}^n, \mathbf{s}^n, \mathbf{x}^n) \mid 1 \leq n \leq N\}$ that do not satisfy constraints (6.15), (6.16), (6.17) at $(\boldsymbol{\alpha}^*, \boldsymbol{\theta}^*, \beta_t^1 = LG_t^1(\boldsymbol{\alpha}^*, \boldsymbol{\theta}^*)(1 - \varepsilon_2))$ for all t , $\beta^2 = LG_2(\boldsymbol{\theta}^*)(1 - \varepsilon_2)$, $\beta^3 = LG_3(\boldsymbol{\alpha}^*)(1 - \varepsilon_2)$
- 2.2. While $\left(\widehat{LG} - \widehat{LB}\right) > \varepsilon$ and $\left(\widehat{LG} - LD_r^{\tilde{n}}\right) > \varepsilon$ Do
- 2.2.1. Solve $LD_r^{\tilde{n}}$ after adding the constraint $\left(\sum_{t=1}^T \beta_t^1\right) + \beta^2 + \beta^3 \geq 0$ to obtain $(\boldsymbol{\alpha}^{\tilde{n}}, \boldsymbol{\theta}^{\tilde{n}}, \beta_t^{1\tilde{n}}, \beta^{2\tilde{n}}, \beta^{3\tilde{n}})$
- 2.2.2. Set $\lambda = \lambda^0$
- 2.2.3. Calculate $\boldsymbol{\alpha}^{\tilde{n}^\lambda}, \boldsymbol{\theta}^{\tilde{n}^\lambda}$: $\boldsymbol{\alpha}^{\tilde{n}^\lambda} = \lambda \boldsymbol{\alpha}^{\tilde{n}} + (1 - \lambda) \hat{\boldsymbol{\alpha}}$,
 $\boldsymbol{\theta}^{\tilde{n}^\lambda} = \lambda \boldsymbol{\theta}^{\tilde{n}} + (1 - \lambda) \hat{\boldsymbol{\theta}}$
- 2.2.4. Solve LG_t^1 with $\boldsymbol{\alpha}^{\tilde{n}^\lambda}, \boldsymbol{\theta}^{\tilde{n}^\lambda}$ for all t ; LG_2 with $\boldsymbol{\theta}^{\tilde{n}^\lambda}$ and LG_3 with $\boldsymbol{\alpha}^{\tilde{n}^\lambda}$ to obtain $(\mathbf{z}^{\tilde{n}^\lambda}, \mathbf{y}^{\tilde{n}^\lambda}, \mathbf{f}^{\tilde{n}^\lambda}, \mathbf{g}^{\tilde{n}^\lambda}, \mathbf{s}^{\tilde{n}^\lambda}, \mathbf{x}^{\tilde{n}^\lambda})$ and $LG(\boldsymbol{\alpha}^{\tilde{n}^\lambda}, \boldsymbol{\theta}^{\tilde{n}^\lambda})$
- 2.2.5. If (any of these problems is infeasible or $\widehat{LG} < 1$) Then LSRP' is infeasible or has zero as optimal objective stop
- 2.2.6. If $\left(LG(\boldsymbol{\alpha}^{\tilde{n}^\lambda}, \boldsymbol{\theta}^{\tilde{n}^\lambda}) < \widehat{LG}\right)$ Then set $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^{\tilde{n}^\lambda}$, $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^{\tilde{n}^\lambda}$, and $\widehat{LG} = LG(\boldsymbol{\alpha}^{\tilde{n}^\lambda}, \boldsymbol{\theta}^{\tilde{n}^\lambda})$
- 2.2.7. Construct a feasible solution $(\bar{\mathbf{z}}^{\tilde{n}^\lambda}, \bar{\mathbf{y}}^{\tilde{n}^\lambda}, \bar{\mathbf{f}}^{\tilde{n}^\lambda}, \bar{\mathbf{g}}^{\tilde{n}^\lambda}, \bar{\mathbf{s}}^{\tilde{n}^\lambda}, \bar{\mathbf{x}}^{\tilde{n}^\lambda})$ from $(\mathbf{z}^{\tilde{n}^\lambda}, \mathbf{y}^{\tilde{n}^\lambda}, \mathbf{f}^{\tilde{n}^\lambda}, \mathbf{g}^{\tilde{n}^\lambda}, \mathbf{s}^{\tilde{n}^\lambda}, \mathbf{x}^{\tilde{n}^\lambda})$ using DH
- 2.2.8. If $\left(\widehat{LB}^{\tilde{n}^\lambda} > \widehat{LB}\right)$ Then, set $(\mathbf{z}^*, \mathbf{y}^*, \mathbf{f}^*, \mathbf{g}^*, \mathbf{s}^*, \mathbf{x}^*) = (\bar{\mathbf{z}}^{\tilde{n}^\lambda}, \bar{\mathbf{y}}^{\tilde{n}^\lambda}, \bar{\mathbf{f}}^{\tilde{n}^\lambda}, \bar{\mathbf{g}}^{\tilde{n}^\lambda}, \bar{\mathbf{s}}^{\tilde{n}^\lambda}, \bar{\mathbf{x}}^{\tilde{n}^\lambda})$ and $\widehat{LB} = \widehat{LB}^{\tilde{n}^\lambda}$
- 2.2.9. If $\left(\left|\widehat{LG} - \widehat{LB}\right| < \varepsilon\right)$ Then go to 3.1
- 2.2.10. If $\left(\left|\widehat{LG} - LD_r^{\tilde{n}}\right| < \varepsilon\right)$ Then go to 3.2
- 2.2.11. If $(\mathbf{z}^{\tilde{n}^\lambda}, \mathbf{y}^{\tilde{n}^\lambda}, \mathbf{f}^{\tilde{n}^\lambda}, \mathbf{g}^{\tilde{n}^\lambda}, \mathbf{s}^{\tilde{n}^\lambda}, \mathbf{x}^{\tilde{n}^\lambda})$ obtained is feasible for $LD_r^{\tilde{n}}$ at $(\boldsymbol{\alpha}^{\tilde{n}}, \boldsymbol{\theta}^{\tilde{n}}, \beta_t^{1\tilde{n}}, \beta^{2\tilde{n}}, \beta^{3\tilde{n}})$ Then
- 2.2.11.1. $\lambda \leftarrow \lambda + \Delta\lambda$, Go to 2.2.3
 Else /* This is the case when the obtained $(\mathbf{z}^{\tilde{n}^\lambda}, \mathbf{y}^{\tilde{n}^\lambda}, \mathbf{f}^{\tilde{n}^\lambda}, \mathbf{g}^{\tilde{n}^\lambda}, \mathbf{s}^{\tilde{n}^\lambda}, \mathbf{x}^{\tilde{n}^\lambda})$ is not feasible for $LD_r^{\tilde{n}}$ at $(\boldsymbol{\alpha}^{\tilde{n}}, \boldsymbol{\theta}^{\tilde{n}}, \beta_t^{1\tilde{n}}, \beta^{2\tilde{n}}, \beta^{3\tilde{n}})$ */
- 2.2.11.2. Include h^{1t}, h^2, h^3 corresponding to $(\mathbf{z}^{\tilde{n}^\lambda}, \mathbf{y}^{\tilde{n}^\lambda}, \mathbf{f}^{\tilde{n}^\lambda}, \mathbf{g}^{\tilde{n}^\lambda}, \mathbf{s}^{\tilde{n}^\lambda}, \mathbf{x}^{\tilde{n}^\lambda})$ in $LD_r^{\tilde{n}}$
-

Figure 6.3. Solution of RLSRP (third part)

-
- 2.2.11.3. Perform N^1 more subgradient iterations in Step 1.2 starting with $\alpha^1 = \alpha^{\tilde{n}^\lambda}$, $\theta^1 = \theta^{\tilde{n}^\lambda}$ and $\mu^1 = 2.0$
- 2.2.11.4. Include h^{1t}, h^2, h^3 corresponding to the elements of $\{(\mathbf{z}^n, \mathbf{y}^n, \mathbf{f}^n, \mathbf{g}^n, \mathbf{s}^n, \mathbf{x}^n) \mid 1 \leq n \leq N^1\}$ that do not satisfy constraints (6.15), (6.16), (6.17) at $(\alpha^{\tilde{n}}, \theta^{\tilde{n}}, \beta_t^{1\tilde{n}}, \beta^{2\tilde{n}}, \beta^{3\tilde{n}})$ in $LD_r^{\tilde{n}}$
- Endif
- 2.2.12. If ($\tilde{n}$ is a multiple of n^d) Then discard rows of $LD_r^{\tilde{n}}$ that are not binding at $(\alpha^{\tilde{n}}, \theta^{\tilde{n}}, \beta_t^{1\tilde{n}}, \beta^{2\tilde{n}}, \beta^{3\tilde{n}})$
- End While

3. Final phase:

3.1. An optimal solution to LSRP' is found stop

3.2.1. Solve RLSRP with h^{1t}, h^2, h^3 of $LD_r^{\tilde{n}}$ after adding h^{1t}, h^2, h^3 corresponding to the solution of LG_t^1 at $\hat{\alpha}, \hat{\theta}$ for all t ; LG_2 at $\hat{\theta}$ and LG_3 at $\hat{\alpha}$ in order to obtain an optimal solution $(d_*^{h^{1t}}, d_*^{h^2}, d_*^{h^3})$ of RLSRP

3.2.2. Construct a feasible solution to LSRP from $(d_*^{h^{1t}}, d_*^{h^2}, d_*^{h^3})$ using PH

3.2.3 If $(\overline{LB} > \widehat{LB})$ Then set $(\mathbf{z}^*, \mathbf{y}^*, \mathbf{f}^*, \mathbf{g}^*, \mathbf{s}^*, \mathbf{x}^*) = (\bar{\mathbf{z}}, \bar{\mathbf{y}}, \bar{\mathbf{f}}, \bar{\mathbf{g}}, \bar{\mathbf{s}}, \bar{\mathbf{x}})$ and $\widehat{LB} = \overline{LB}$

Figure 6.4. Solution of RLSRP (fourth part)

$\mathbf{x}, \mathbf{s}, \mathbf{z}, \mathbf{y}$. Besides, the algorithm in Figure 6.1 when employed in solving RLSRP with equality constraints on some components of $\mathbf{x}^f, \mathbf{s}^f, \mathbf{z}^f, \mathbf{y}^f$ can be improved by using the parent node's solution, i.e. the node from which the new node is obtained with a single equality constraint on a component of $\mathbf{x}^f, \mathbf{s}^f, \mathbf{z}^f, \mathbf{y}^f$. h^{1t}, h^2, h^3 corresponding to positive d_*^{1t}, d_*^2, d_*^3 in the optimal solution of the parent node's problem can be evaluated. Among them, those that satisfy the additional constraint defining the new node can be found. Problem LD can be solved with these h^{1t}, h^2, h^3 after adding the constraint $\sum_{t=1}^T \beta_t^1 + \beta^2 + \beta^3 \geq 0$. Setting α^1, θ^1 to α and θ values obtained with this procedure

```

1. Set  $(\bar{\mathbf{z}}, \bar{\mathbf{y}}, \bar{\mathbf{f}}, \bar{\mathbf{g}}, \bar{\mathbf{s}}, \bar{\mathbf{x}}) = (\mathbf{z}^n, \mathbf{y}^n, \mathbf{f}^n, \mathbf{g}^n, \mathbf{s}^n, \mathbf{x}^n)$ 
2. If  $(\bar{z}_{ikt} = 1$  and  $\bar{x}_{ik} = 0$  for a sensor  $(i, k)$  in period  $t$  with  $\bar{y}_t = 1)$  Then Set  $\bar{y}_t = 0$ 
3. For all period  $t$  with  $\bar{y}_t = 1$  Do
3.1.   For all sensor  $(i, k)$  Do
3.1.1.   For all point  $j$  with  $b_{ijk} = 1$  Do
3.1.1.1.   If  $(\bar{g}_{ikjt} > 0$  and  $\bar{s}_j \neq 1)$  Then
3.1.1.1.1.   Try to route  $\bar{g}_{ikjt}$  data packets to the sinks within the transmission
               range of sensor  $(i, k)$  without exceeding flow capacity  $v_k$ 
3.1.1.1.2.   Set  $\bar{g}_{ikjt} = 0$ , and add new flow quantities to the corresponding
               components of  $\bar{\mathbf{g}}$ 
3.1.1.1.3.   If  $\bar{g}_{ikjt}$  data packets cannot be routed to sinks Then set  $\bar{y}_t = 0$ 
               End If
           End For
       End For
   End For
4.1. While  $(\sum_{t=1}^T \bar{\gamma}_{ikt} \bar{y}_t > E_k$  for at least a sensor  $(i, k)$  with  $\bar{x}_{ik} = 1)$  Do
4.1.1.   For all period  $t$  with  $\bar{y}_t = 1$  Do
4.1.1.1.   Set  $\sigma_t = 0$ 
4.1.1.2.   For all sensor  $(i, k)$  with  $\bar{x}_{ik} = 1$  Do
4.1.1.2.1.   If  $(\left(\sum_{t=1}^T \bar{\gamma}_{ikt} \bar{y}_t > E_k\right)$  and  $\bar{\gamma}_{ikt} > 0)$  Then  $\sigma_t \leftarrow \sigma_t + 1$ 
               End For
           End For
4.1.2.   Set  $\bar{y}_{t^*} = 0$  for  $t^* = \arg \max_{\{t: \bar{y}_t = 1\}} \sigma_t$ 
   End While

```

Figure 6.5. Algorithm dual heuristic

accelerates the convergence of the algorithm in Figure 6.1. In order to improve the algorithm in Figure 6.1 in this way, for each node i in *node list*, h^{1t}, h^2, h^3 corresponding to positive d_*^{1t}, d_*^2, d_*^3 in the optimal solution of the node's problem should be stored,

-
1. Choose the sink locations of $h^2 = \arg \max_{h^2 \in H_n^2} d_*^{h^2}$ as the final sink locations $\bar{\mathbf{s}}$
 2. Choose sensor locations of $h^3 = \arg \max_{h^3 \in H_n^3} d_*^{h^3}$ as the final sensor locations $\bar{\mathbf{x}}$
 3. For all period t Do
 - 3.1. Find all $h^{1t} \in H_n^{1t}$ with $y_t^{h^{1t}} = 1$, $\mathbf{z}^{h^{1t}} \leq \bar{\mathbf{x}}$, and $\mathbf{g}^{h^{1t}}$ satisfying constraint (5.13) after the application of the third step of DH
 - 3.2. Among them choose the one with the highest $d_*^{h^{1t}}$ value as the active sensor set of period t
 - 3.3. If there is no $h^{1t} \in H_n^{1t}$ as described in Step 3.1 Then Set $\bar{\mathbf{y}}_t = 0$
 - End For
 4. Apply Step 4 of DH to obtain the final feasible solution satisfying the energy Constraints (6.2)
-

Figure 6.6. Algorithm primal heuristic

in addition to w_i and the fixed variables defining the node. In selecting the branching variable ν , first, fractional components of $\mathbf{x}^f$ are evaluated. Among them, fractional component $\mathbf{x}_{ik}^f$ that makes the greatest number of $h^3 \in H^3$ infeasible in the parent node when restricted to be equal to $\delta(x_{ik}^f)$ is selected. Here, $\delta(\xi)$ is a function that is equal to 0 if ξ is greater than or equal to 0.5, it is 1 otherwise. This branching rule is similar to that presented in Klose and Görtz [65]. If $\mathbf{x}^f$ is an integer vector, variable $\mathbf{s}^f$ is considered. Fractional component $\mathbf{s}_i^f$ that makes the greatest number of $h^2 \in H^2$ infeasible in the father node when restricted to be equal to $\delta(\mathbf{s}_i^f)$ is selected. If both $\mathbf{x}^f$ and $\mathbf{s}^f$ are integer vectors, variable $\mathbf{z}^f$ is considered. Fractional component $\mathbf{z}_{ikt}^f$ that makes the greatest number of $h^{1t} \in H^{1t}$ infeasible in the father node when restricted to be equal to $\delta(\mathbf{z}_{ikt}^f)$ is selected. If all $\mathbf{x}^f, \mathbf{s}^f, \mathbf{z}^f$ are integer vectors, fractional component $\mathbf{y}_t^f$ that makes the largest $h^{1t} \in H^{1t}$ infeasible in the parent node when restricted to be equal to $\delta(\mathbf{y}_t^f)$ is selected for further branching. Notice that we prefer branching variables $\mathbf{x}^f, \mathbf{s}^f, \mathbf{z}^f, \mathbf{y}^f$ of LSRP to branching variables $d^{h^{1t}}, d^{h^2}, d^{h^3}$ of RLSRP because they do not alter the main building blocks of the algorithm in Figure 6.1. If LSRP' corresponding to a node is infeasible or has zero as optimal objective, then, that node

is removed from *node list* in step 3.6.2.1. On the other hand, if $\mathbf{x}^f, \mathbf{s}^f, \mathbf{z}^f, \mathbf{y}^f$ vectors obtained from the solution of a node's problem are all integer, then, L_{LB} is updated if $\widehat{LG}$ is higher than L_{LB} . And the node is removed from *node list* as described in Step 3.6.2.4. Finally, $\max_{i \in \text{node list}} w_i$ is equal to L_{UB} , and it is updated if there is a change in its value at Step 3.7.

1. Solve RLSRP using the algorithm in Figure 6.1.
 - 1.1. If an optimal solution to LSRP' is found Then STOP
 - 1.2. If LSRP' is infeasible or has zero as optimal objective Then STOP
 2. Set $L_{LB} = LB^*$, $(\mathbf{z}_{LB}^*, \mathbf{y}_{LB}^*, \mathbf{f}_{LB}^*, \mathbf{g}_{LB}^*, \mathbf{s}_{LB}^*, \mathbf{x}_{LB}^*) = (\mathbf{z}^*, \mathbf{y}^*, \mathbf{f}^*, \mathbf{g}^*, \mathbf{s}^*, \mathbf{x}^*)$, $L_{UB} = \widehat{LG}$, *node list* = {0}, $w_0 = LG^*$
 3. While *node list* $\neq \emptyset$ Do
 - 3.1. Remove all nodes i with $w_i \leq L_{LB}$ from *node list*
 - 3.2. Select the node $i^* = \arg\max_{i \in \text{node list}} w_i$
 - 3.3. Remove node i^* from *node list*
 - 3.4. Select the branching variable ν
 - 3.5. Add nodes corresponding to $\nu = 0, \nu = 1$ to *node list*
as nodes $|\text{node list}| + 1, |\text{node list}| + 2$
 - 3.6. For $i \in (|\text{node list}| + 1, |\text{node list}| + 2)$ Do
 - 3.6.1. Solve RLSRP that corresponds to node i using the Algorithm
in Figure 6.1 considering variables having fixed value defining node i
 - 3.6.2. If LSRP' that corresponds to node i is infeasible or has zero
as optimal objective Then
 - 3.6.2.1. Remove node i from *node list*
 - Else
-

Figure 6.7. Branch and price (first part)

-
- 3.6.2.2. $w_i = \widehat{LG}$
- 3.6.2.3. If $(\widehat{LB} > L_{LB})$ Then set $L_{LB} = \widehat{LB}$,
 $(\mathbf{z}_{LB}^*, \mathbf{y}_{LB}^*, \mathbf{f}_{LB}^*, \mathbf{g}_{LB}^*, \mathbf{s}_{LB}^*, \mathbf{x}_{LB}^*) \leftarrow (\mathbf{z}^*, \mathbf{y}^*, \mathbf{f}^*, \mathbf{g}^*, \mathbf{s}^*, \mathbf{x}^*)$
- 3.6.2.4. If $(\mathbf{x}^f, \mathbf{s}^f, \mathbf{z}^f, \mathbf{y}^f)$ (6.27) obtained from the solution of the problem is integer Then
- 3.6.2.4.1. If $(\widehat{LG} > L_{LB})$ Then set $L_{LB} = \widehat{LG}$,
 $(\mathbf{z}_{LB}^*, \mathbf{y}_{LB}^*, \mathbf{f}_{LB}^*, \mathbf{g}_{LB}^*, \mathbf{s}_{LB}^*, \mathbf{x}_{LB}^*) = (\mathbf{z}^f, \mathbf{y}^f, \mathbf{f}^f, \mathbf{g}^f, \mathbf{s}^f, \mathbf{x}^f)$
- 3.6.2.4.2. Remove node i from *node list*
- End If
- End If
- End For
- 3.7. If $(\max_{i \in \text{node list}} w_i < L_{UB})$ Then Set $L_{UB} = \max_{i \in \text{node list}} w_i$
- 3.8. If $(L_{UB} < 1)$ Then LSRP' is infeasible or has zero
as optimal objective stop
- 3.9. If $(|L_{UB} - L_{LB}| < \varepsilon)$ Then $(\mathbf{z}_{LB}^*, \mathbf{y}_{LB}^*, \mathbf{f}_{LB}^*, \mathbf{g}_{LB}^*, \mathbf{s}_{LB}^*, \mathbf{x}_{LB}^*)$ is an optimal solution
of LSRP' STOP
- End While
-

Figure 6.8. Branch and price (second part)

7. A LAGRANGEAN HEURISTIC

7.1. Introduction

As can be seen from the computational results reported in Chapter 10 the exact solution procedure proposed in the Chapter 6 is not efficient in solving large LSRP instances. Therefore, heuristic solution procedures are needed. In this chapter³ we propose a Lagrangean heuristic for LSRP. We relax complicating constraints using Lagrangean multipliers, and solve the Lagrangean dual problem using the subgradient algorithm. We construct feasible solutions of LSRP from the solutions of the Lagrangean problems that do not necessarily satisfy the relaxed constraints.

7.2. Lagrangean Heuristic

First we define new nonnegative continuous variables $H_{ikt} = h_{ik}z_{ikt}$ and $D_{ikt} = e_{ik}^s z_{ikt}$ for all i, k, t and obtain the following equivalent formulation of LSRP

$$\max \quad L \tag{7.1}$$

s.t.

$$H_{ikt} = h_{ik}z_{ikt} \quad \text{for all } i, k, t \tag{7.2}$$

$$D_{ikt} = e_{ik}^s z_{ikt} \quad \text{for all } i, k, t \tag{7.3}$$

$$\text{Constraints (5.5) where } h_{ik}z_{ikt} \text{ is replaced by } H_{ikt} \tag{7.4}$$

$$\text{Constraints (5.6) where } e_{ik}^s z_{ikt} \text{ is replaced by } D_{ikt} \tag{7.5}$$

and constraints (5.2)–(5.4), (5.8)–(5.14).

When constraints (5.3), (5.10)–(5.13), (7.2), (7.3), and (5.8) are relaxed with Lagrange multipliers $\mu \geq 0$, $\theta^1 \geq 0$, $\theta^2 \geq 0$, $\theta^3 \geq 0$, $\theta^4 \geq 0$, α^1 , α^2 , and multiplier $\lambda \geq 0$,

³The paper Türkoğulları et al. [68] including the results of this chapter appeared in the Journal of the Operational Research Society.

respectively, the following Lagrangean problem is obtained:

$$\begin{aligned}
L_{UB}(\mu, \lambda, \boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \boldsymbol{\theta}^3, \boldsymbol{\theta}^4, \boldsymbol{\alpha}^1, \boldsymbol{\alpha}^2) = \max \quad & L + \sum_{t=1}^T \sum_{j=1}^J \mu_{jt} \left\{ \left(\sum_{i=1}^J \sum_{k=1}^K a_{ijk} z_{ikt} \right) - d_j y_t \right\} \\
& + \lambda \left\{ B - \sum_{k=1}^K \sum_{i=1}^J c_{ik} x_{ik} \right\} + \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{l=1}^K \sum_{t=1}^T b_{ijk} \theta_{ikjl}^1 (u_{kl} z_{ikt} - f_{ikjl}) \\
& \quad \quad \quad j \neq i \\
& + \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{l=1}^K \sum_{t=1}^T b_{ijk} \theta_{ikjl}^2 (u_{kl} z_{jlt} - f_{ikjl}) + \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{t=1}^T b_{ijk} \theta_{ikjt}^3 (v_k z_{ikt} - g_{ikjt}) \\
& \quad \quad \quad j \neq i \\
& + \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{t=1}^T b_{ijk} \theta_{ikjt}^4 (v_k s_j - g_{ikjt}) + \sum_{i=1}^J \sum_{k=1}^K \sum_{t=1}^T \alpha_{ikt}^1 (H_{ikt} - h_{ik} z_{ikt}) \\
& + \sum_{i=1}^J \sum_{k=1}^K \sum_{t=1}^T \alpha_{ikt}^2 (D_{ikt} - e_{ik}^s z_{ikt}) \tag{7.6}
\end{aligned}$$

$$\text{s.t. (5.2), (5.4), (5.7), (5.9), (5.14), (7.4), (7.5).} \tag{7.7}$$

The Lagrangean problem decomposes into four subproblems for a given set of Lagrange multiplier values. We explain the solution approach of each subproblem below where the optimal value of each subproblem for given values of Lagrange multipliers is denoted by Z . Actually, Z is a function of the corresponding Lagrange multipliers, but in order to simplify the notation we drop them (e.g., $Z_1 = Z_1(\mu)$).

7.2.1. First Subproblem

The first subproblem SP1 is a mathematical program given as

$$\text{SP1:} \quad Z_1 = \max \quad L - \sum_{j=1}^J \sum_{t=1}^T \mu_{jt} d_j y_t \tag{7.8}$$

$$\text{s.t. (5.2), (5.4)}$$

$$y_t \in \{0, 1\} \quad \text{for all } t. \tag{7.9}$$

The objective function can be written as $L - \sum_{t=1}^T \sigma_t y_t$ where $\sigma_t = \sum_{j=1}^J \mu_{jt} d_j$. Here the decision variables are L and $y_t, t = 1, \dots, T$. Notice that if the value of L is known, then optimal y_t values can easily be determined. Suppose that $L = t_0 - 1$. Then, $y_t = 1$ for $1 \leq t < t_0$ due to constraints (5.4). Furthermore, since σ_t is nonnegative, $y_t = 0$ for

$t_0 \leq t \leq T$ to yield the maximum possible objective value. As a result, we can try all possible values for L , and select the one which provides the highest objective value Z_1 .

Proposition 15. *The computational complexity of this solution procedure is $\mathcal{O}(JT)$*

Proof. Calculating σ_t is $\mathcal{O}(J)$. There are T σ_t values to calculate. Therefore calculating all σ_t is $\mathcal{O}(JT)$. Calculating all objective values for each value of L is $\mathcal{O}(T)$. Finding L value which gives the maximum objective value is $\mathcal{O}(T)$. Therefore computational complexity of the solution procedure is $\mathcal{O}(JT)$. Q.E.D.

7.2.2. Second Subproblem

The second subproblem is given as

$$\begin{aligned} \text{SP2:} \quad Z_2 = \max \quad & \sum_{t=1}^T \sum_{i=1}^J \sum_{k=1}^K \gamma_{ikt} z_{ikt} - \lambda \sum_{k=1}^K \sum_{i=1}^J c_{ik} x_{ik} \\ \text{s.t.} \quad & (5.7) \end{aligned} \tag{7.10}$$

$$x_{ik}, z_{ikt} \in \{0, 1\} \quad \text{for all } i, k, t \tag{7.11}$$

where

$$\begin{aligned} \gamma_{ikt} = & \left(\sum_{j=1}^J \mu_{jt} a_{ijk} \right) + \left(\sum_{\substack{j=1 \\ j \neq i}}^J \sum_{l=1}^K b_{ijk} u_{kl} \theta_{ikjl}^1 \right) + \left(\sum_{\substack{j=1 \\ j \neq i}}^J \sum_{l=1}^K b_{jil} u_{lk} \theta_{jlik}^2 \right) \\ & + \left(\sum_{j=1}^J b_{ijk} v_k \theta_{ikjt}^3 \right) - \alpha_{ikt}^1 h_{ik} - \alpha_{ikt}^2 e_{ik}^s. \end{aligned} \tag{7.12}$$

Note that SP2 decomposes with respect to sensor type k and point i . That is, $Z_2 = \sum_{i=1}^J \sum_{k=1}^K Z_2^{ik}$ where Z_2^{ik} is obtained by solving problem SP2^{ik} given as

$$\begin{aligned} \text{SP2}^{ik}: \quad Z_2^{ik} = \max \quad & \sum_{t=1}^T \gamma_{ikt} z_{ikt} - \lambda c_{ik} x_{ik} \\ \text{s.t.} \quad & (5.7), (7.11). \end{aligned} \tag{7.13}$$

Note that if a type- k sensor is not located at point i (i.e. $x_{ik} = 0$), then it cannot be activated either (i.e. $z_{ikt} = 0, t = 1, \dots, T$). This gives an objective value of $Z_2^{ik} = 0$. If there is a type- k sensor at point i , this sensor has to be activated in those periods t for which $\gamma_{ikt} > 0$ because of the maximization objective. Since the cost of placing a sensor is λc_{ik} , a type- k sensor is placed at point i if $\sum_{t:\gamma_{ikt}>0} \gamma_{ikt} > \lambda c_{ik}$. If this is the case, then $x_{ik} = 1$, $z_{ikt} = 1$ for $\{t : \gamma_{ikt} > 0\}$ and $z_{ikt} = 0$ for the remaining t values. Otherwise $x_{ik} = 0$, $z_{ikt} = 0$ for all periods $t = 1, \dots, T$.

Proposition 16. *The computational complexity of this solution procedure is $\mathcal{O}(\tau^{s_1} JKT + \tau^{c_1} JK^2T + \tau^{c_2} JKT)$, where $\tau^{s_1} = \max_{i,k} \left\{ \sum_{j=1}^J a_{ijk} \right\}$ is the maximum number of points in the sensor field that are within the sensing range of a sensor, $\tau^{c_1} = \max_{i,k} \left\{ \sum_{j=1}^J b_{ijk} \right\}$ is the maximum number of points in the sensor field that are within the communication range of a sensor, and $\tau^{c_2} = \max_j \left\{ \sum_{i=1}^J \sum_{k=1}^K b_{ijk} \right\}$ is the maximum number of sensors having a point of the sensor field within their communication range.*

Proof. For each sensor of type- k placed at site i calculating all γ_{ikt} is $\mathcal{O}(\tau^{s_1}T + \tau^{c_1}KT + \tau^{c_2}KT)$. Comparison with zero and finding the objective value is $\mathcal{O}(T)$. This calculation is repeated for JK sensors. Q.E.D.

7.2.3. Third Subproblem

The third subproblem is given as follows:

$$\begin{aligned}
 \text{SP3: } Z_3 = \max \quad & \sum_{i=1}^J \sum_{k=1}^K \sum_{t=1}^T \alpha_{ikt}^1 H_{ikt} + \sum_{i=1}^J \sum_{k=1}^K \sum_{t=1}^T \alpha_{ikt}^2 D_{ikt} - \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{l=1}^K \sum_{t=1}^T b_{ijk} \theta_{ikjlt}^1 f_{ikjlt} \\
 & - \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{l=1}^K \sum_{t=1}^T b_{ijk} \theta_{ikjlt}^2 f_{ikjlt} - \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{t=1}^T b_{ijk} \theta_{ikjlt}^3 g_{ikjlt} \\
 & - \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{t=1}^T b_{ijk} \theta_{ikjlt}^4 g_{ikjlt} \quad (7.14)
 \end{aligned}$$

s.t.

$$(7.4), (7.5)$$

$$H_{ikt}, D_{ikt}, f_{ikjlt}, g_{ikjlt} \geq 0. \quad (7.15)$$

Note that SP3 is an LP. Hence, its solution can be obtained by means of commercial LP solvers like CPLEX [62].

Proposition 17. *Based on Karmarkar's interior point algorithm [60], the solution algorithm is of complexity $\mathcal{O}((\tau^{c_1})^3 J^3 K^6 T^3 L_{B_1})$ where L_{B_1} is the size of the LP instance in terms of the number of bits necessary for storage.*

Proof. Number of variables of the LP is $\mathcal{O}(\tau^{c_1} JK^2 T)$. Number of constraints is $\mathcal{O}(JKT)$. Q.E.D.

7.2.4. Fourth Subproblem

The forth subproblem can be written as

$$\text{SP4:} \quad Z_4 = \max \sum_{j=1}^J \delta_j s_j \quad (7.16)$$

$$\text{s.t. (5.9)}$$

$$s_j \in \{0, 1\} \quad (7.17)$$

where

$$\delta_j = \sum_{i=1}^J \sum_{k=1}^K \sum_{t=1}^T \theta_{ikjt}^4 v_k b_{ijk}. \quad (7.18)$$

Recall that $s_j = 1$ if there is a sink at point j , and it is zero otherwise. To solve SP4, we first sort coefficients δ_j in nonincreasing order as $\delta_{j[1]} \geq \delta_{j[2]} \geq \dots \geq \delta_{j[J]}$. Maximizing the objective function subject to Constraint (5.9) corresponds to locating sinks at points $j[1], j[2], \dots, j[S]$. In other words, $s_{j[m]} = 1$ for $m = 1, \dots, S$ and $s_{j[m]} = 0$ for $m = S + 1, \dots, J$.

Proposition 18. *The computational complexity of this solution procedure is $\mathcal{O}(\tau^{c_2} JT + J \log J)$.*

Proof. Calculating all δ_j is $\mathcal{O}(\tau^{c_2} JT)$. Sorting δ_j values is $\mathcal{O}(J \log J)$. Q.E.D.

The value $L_{UB} = Z_1 + Z_2 + Z_3 + Z_4 + \lambda B$ is an upper bound on the optimal value of the original problem for any Lagrange multiplier vector set $\{\mu, \lambda, \boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \boldsymbol{\theta}^3, \boldsymbol{\theta}^4, \boldsymbol{\alpha}^1, \boldsymbol{\alpha}^2\}$. To find the best (smallest) upper bound, we solve the Lagrangean dual problem

$$L_{UB}^* = \min_{\mu \geq 0, \lambda \geq 0, \boldsymbol{\theta}^1 \geq 0, \boldsymbol{\theta}^2 \geq 0, \boldsymbol{\theta}^3 \geq 0, \boldsymbol{\theta}^4 \geq 0, \boldsymbol{\alpha}^1, \boldsymbol{\alpha}^2} L_{UB}(\mu, \lambda, \boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \boldsymbol{\theta}^3, \boldsymbol{\theta}^4, \boldsymbol{\alpha}^1, \boldsymbol{\alpha}^2) \quad (7.19)$$

by using a subgradient optimization algorithm. At each iteration n of the subgradient optimization procedure, the current upper bound $L_{UB}^{(n)}$ is obtained by solving (opti-

mally) the subproblems SP1, SP2, SP3, and SP4. Then, Lagrange multipliers $\mu^{(n)}$, $\lambda^{(n)}$, $\theta^1^{(n)}$, $\theta^2^{(n)}$, $\theta^3^{(n)}$, $\theta^4^{(n)}$, $\alpha^1^{(n)}$, $\alpha^2^{(n)}$ are updated using the best available lower bound L_{LB}^* . Lower bounds $L_{LB}^{(n)}$ are computed at each iteration n by constructing a feasible solution of the original problem from the solution of the Lagrangean problem using the algorithm in Figure 7.3 which will be explained later. The output of the Lagrangean Heuristic (LH) is the best feasible solution with objective value L_{LB}^* and the best (smallest) upper bound L_{UB}^* found throughout the iterations. Different criteria can be considered to terminate the iterations. One possibility is to stop the subgradient algorithm if $L_{UB}^* - L_{LB}^* < \varepsilon_1$ where ε_1 is a small positive constant. However, if there is a duality gap larger than ε_1 , this can never be possible. Another termination criterion is based on the step size parameter π . If there is no improvement in the best upper bound L_{UB}^* during N consecutive iterations, the value of π is halved. $N = 20$ is suggested in the literature [69]. When π becomes smaller than a threshold ε_2 , the algorithm is terminated. Our experience with Lagrangean heuristics reveals that in the majority of the problems good feasible solutions can be found quickly although the best upper bound for a maximization problem or the best lower bound for a minimization problem do not improve too much when the number of iterations exceed a certain threshold value. For this reason, we also set a limit called *iterlim* for the total number of iterations. The steps of LH are given in Figure 7.1.

7.2.5. Generating a Feasible Solution

As mentioned before, at each iteration, LH requires the construction of a feasible solution from the solution of the Lagrangean problem. We use the algorithm in Figure 7.3 for this purpose whose main idea is to recover the infeasibilities that occur with respect to the relaxed constraints. It uses three procedures given as Algorithm in Figure 7.5, Algorithm in Figure 7.7 and Algorithm in Figure 7.8.

Proposition 19. *Computational complexity of the algorithm in Figure 7.5 is given as*

$$\mathcal{O}(\tau^{s_1}\tau^{s_2}JKT + J^2K^2T^2 + \tau^{s_1}J^2K^2T + J^2K^2 \log(JK)T), \quad (7.20)$$

-
1. Initialization: Set iteration counter $n = 0$, $\pi^{(0)} = 2$, $L_{UB}^* = \infty$, $L_{LB}^* = 0$, and $\mu_{jt}^{(n)}, \lambda^{(n)}, \alpha_{ikt}^{1(n)}, \alpha_{ikt}^{2(n)}, \theta_{ikjlt}^{1(n)}, \theta_{ikjlt}^{2(n)}, \theta_{ikjt}^{3(n)}, \theta_{ikjt}^{4(n)} = 1$ for all i, k, j, l, t ;
 2. While $L_{UB}^* - L_{LB}^* \geq \varepsilon_1$ and $\pi \geq \varepsilon_2$ and $n \leq iterlim$ Do
 3. Solve subproblems SP1, SP2, SP3, SP4,
 compute $L_{UB}^{(n)} = Z_1^{(n)} + Z_2^{(n)} + Z_3^{(n)} + Z_4^{(n)} + \lambda B$
 and update $L_{UB}^* = \min \{L_{UB}^*, L_{UB}^{(n)}\}$
 4. If L_{UB}^* is not updated during N iterations Then set $\pi \leftarrow \pi/2$
 5. Construct a feasible solution with objective value $L_{LB}^{(n)}$ using the algorithm in Figure 7.3 and update $L_{LB}^* = \max \{L_{LB}^*, L_{LB}^{(n)}\}$.
 6. Update Lagrange multipliers $\mu, \lambda, \theta^1, \theta^2, \theta^3, \theta^4, \alpha^1, \alpha^2$:

$$\mu_{jt}^{(n+1)} = \max \left\{ \mu_{jt}^{(n)} - \eta^{(n)} \left(\left(\sum_{i=1}^J \sum_{k=1}^K a_{ijk} z_{ikt} \right) - d_j y_t \right), 0 \right\}$$

$$\lambda^{(n+1)} = \max \left\{ \lambda^{(n)} - \eta^{(n)} \left\{ B - \sum_{k=1}^K \sum_{i=1}^J c_{ik} x_{ik} \right\}, 0 \right\}$$

$$\alpha_{ikt}^{1(n+1)} = \alpha_{ikt}^{1(n)} - \eta^{(n)} (H_{ikt} - h_{ik} z_{ikt})$$

$$\alpha_{ikt}^{2(n+1)} = \alpha_{ikt}^{2(n)} - \eta^{(n)} (D_{ikt} - e_{ik}^s z_{ikt})$$

$$\theta_{ikjlt}^{1(n+1)} = \max \left\{ \theta_{ikjlt}^{1(n)} - b_{ijk} \eta^{(n)} (u_{kl} z_{ikt} - f_{ikjlt}), 0 \right\}$$

$$\theta_{ikjlt}^{2(n+1)} = \max \left\{ \theta_{ikjlt}^{2(n)} - b_{ijk} \eta^{(n)} (u_{kl} z_{jlt} - f_{ikjlt}), 0 \right\}$$

$$\theta_{ikjt}^{3(n+1)} = \max \left\{ \theta_{ikjt}^{3(n)} - b_{ijk} \eta^{(n)} (v_k z_{ikt} - g_{ikjt}), 0 \right\}$$

$$\theta_{ikjt}^{4(n+1)} = \max \left\{ \theta_{ikjt}^{4(n)} - b_{ijk} \eta^{(n)} (v_k s_j - g_{ikjt}), 0 \right\}$$
 where $\eta^{(n)} = \frac{\pi(L_{UB}^{(n)} - L_{LB}^*)}{A}$ with
-

Figure 7.1. Lagrangean Heuristic (first part)

where $\tau^{s_1} = \max_{i,k} \left\{ \sum_{j=1}^J a_{ijk} \right\}$ is the maximum number of points in the sensor field that a sensor can cover. And, $\tau^{s_2} = \max_j \left\{ \sum_{i=1}^J \sum_{k=1}^K a_{ijk} \right\}$ is the maximum number of sensors covering a point of the sensor field.

Proof. In each period t closing redundant sensors is $\mathcal{O}(\tau^{s_1} \tau^{s_2} JK)$. Finding uncoverage is $\mathcal{O}(\tau^{s_2} J)$. We repeat at worst JK times the operations of finding the sensor with P^* or R^* ($\mathcal{O}(JKT + \tau^{s_1} JK)$) and removing sensors ($\mathcal{O}(JKT + JK \log(JK))$). Q.E.D.

$$\begin{aligned}
A = & \sum_{t=1}^T \sum_{j=1}^J \left[\left(\sum_{i=1}^J \sum_{k=1}^K a_{ijk} z_{ikt} \right) - d_j y_t \right]^2 + \left(B - \sum_{k=1}^K \sum_{i=1}^J c_{ik} x_{ik} \right)^2 + \\
& \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{l=1}^K \sum_{t=1}^T b_{ijk} (u_{kl} z_{ikt} - f_{ikjlt})^2 + \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{l=1}^K \sum_{t=1}^T b_{ijk} (u_{kl} z_{jlt} - f_{ikjlt})^2 + \\
& + \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{t=1}^T b_{ijk} (v_k z_{ikt} - g_{ikjt})^2 + \sum_{i=1}^J \sum_{k=1}^K \sum_{j=1}^J \sum_{t=1}^T b_{ijk} (v_k s_j - g_{ikjt})^2 + \\
& + \sum_{i=1}^J \sum_{k=1}^K \sum_{t=1}^T (H_{ikt} - h_{ik} z_{ikt})^2 + \sum_{i=1}^J \sum_{k=1}^K \sum_{t=1}^T (D_{ikt} - e_{ik}^s z_{ikt})^2
\end{aligned}$$

7. $n \leftarrow n + 1$

8. End While

Figure 7.2. Lagrangean Heuristic (second part)

1. For each type- k sensor at point i Do

If $\sum_{t=1}^T (e_{ik}^s + h_{ik} e_k^c) z_{ikt} > E_k$ Then

While $\sum_{t=1}^T (e_{ik}^s + h_{ik} e_k^c) z_{ikt} > E_k$ Do

$\bar{t} = \max_{1 \leq t \leq T} \{t : z_{ikt} = 1\}$

$z_{ikt\bar{t}} = 0$

End While

End If

End For

2. Use the algorithm in Figure 7.5 to obtain a feasible solution subject to the budget constraint and coverage constraints with objective value L'

Figure 7.3. Generation of a feasible solution (first part)

Proposition 20. *Computational complexity of the algorithm in Figure 7.7 is given as $\mathcal{O}(\tau^{c_2} SJ + JK)$, where $\tau^{c_2} = \max_j \left\{ \sum_{i=1}^J \sum_{k=1}^K b_{ijk} \right\}$ is the maximum number of sensors having a point of the sensor field within their communication range.*

Proof. Initialization is $\mathcal{O}(JK)$. We repeat S times the operations of computing

```

3. If  $L' = 0$  Then
    Stop,
Else
    use the algorithm in Figure 7.7 to locate the sinks
End If
4. Set  $t = 1$ 
   While  $t \leq L'$  Do
       If the network is connected in  $t$  Then
            $t \leftarrow t + 1$ 
       Else
           use the algorithm in Figure 7.8 to restore connectivity
           If connectivity is obtained Then
                $L'' = t, t \leftarrow t + 1$ 
           Else
                $L'' = t - 1$  and Go to 5
           End If
       End If
   End While
5. Set  $\hat{L} = L''$ 
   Solve the LP described at the end of this subsection to determine data
   flows  $\mathbf{f}$  and  $\mathbf{g}$ 
   If a feasible solution is found, it is optimal with  $L_{LB} = L''$  Then
       Stop
   Else
        $L'' \leftarrow L'' - 1$ , Go to 5
   End If

```

Figure 7.4. Generation of a feasible solution (second part)

-
1. $t = 1$
 2. While $t \leq L_{UB}^{(n)}$ Do
 3. Close all sensors redundant for coverage
 4. If every point is covered Then
 - If the budget constraint is satisfied Then
 - $L' \geq t, t \leftarrow t + 1$
 - Else
 - Remove unused sensors in $[0, t]$ starting with those having the largest cost until the budget constraint is satisfied.
 - If this is not possible Then
 - $L' = t - 1$
 - Stop
 - End If
 - End If
-

Figure 7.5. Restoring coverage and budget constraints feasibility (first part)

$no_cov(j)$ for all j ($\mathcal{O}(\tau^{c_2}J)$) and updating $\bar{x}$ ($\mathcal{O}(\tau^{c_2})$). Q.E.D.

Proposition 21. *Computational complexity of the algorithm in Figure 7.8 is $\mathcal{O}(\tau^{c_2}J^2K^3)$.*

Proof. We repeat at worst JK times the following operations. Finding all $no_con1(i, k, t)$ or all $no_con2(i, k, t)$ is $\mathcal{O}(\tau^{c_2}JK^2)$. Finding the maximum argument is $\mathcal{O}(JK)$. Checking connectivity is $\mathcal{O}(\tau^{c_2}K + KJ)$. Q.E.D.

In Step 1, we deal with the infeasibility due to the sensors which remain active longer than their battery energy allows. More specifically, if a type- k sensor at point i is active in more than $\lfloor E_k / (e_{ik}^s + h_{ik}e_k^c) \rfloor$ periods, i.e. constraint $\sum_{t=1}^T (e_{ik}^s + h_{ik}e_k^c)z_{ikt} \leq E_k$ is violated, then this sensor is switched from active mode to standby mode starting from the last period in which the sensor is active until the constraint is satisfied. This

```

5. Else /* every point is not covered */
    Find the sensor with the highest  $P_{ikt}$  value,  $P^*$ 
    If  $P^* > 0$  Then
        activate that sensor
    Else
        find the sensor with the highest  $R_{ikt}$  value  $R^*$ 
        If  $R^* = 0$  Then
             $L' = t - 1$  and stop
        Else
            If budget is enough Then
                activate the sensor
            Else
                remove unused sensors in  $[0, t]$  starting with those
                having the largest cost until the budget constraint is
                satisfied.
            If this is not possible Then
                 $L' = t - 1$ 
                Stop
            End If
        End If
    End If
End If
End While

```

Figure 7.6. Restoring coverage and budget constraints feasibility (second part)

means that some z_{ikt} variables are set to zero. The next step is to make adjustments in the values of the decision variables x_{ik} and z_{ikt} so that the solution satisfies coverage constraints (5.3) and budget constraint (5.8). This is accomplished in Step 2 by calling the algorithm in Figure 7.5 for each period $t = 1, \dots, L_{UB}^{(n)}$.

```

1. Initialization: If there is a sensor at point  $i$  Then
    Set  $\bar{x}_i = 1$ 
Else
    Set  $\bar{x}_i = 0$ 
End If

 $n = 1$ 

2. While  $n < S$  Do
    Compute  $no\_cov(j) = \sum_{i=1}^J b_{ijk_i} \bar{x}_i$  for  $j = 1, \dots, J$ 
     $j^* = \arg \max_j \{no\_cov(j)\}$ 
    Locate a sink at  $j^*$ 
     $\bar{x}_i = 0$  for  $i$  such that  $\bar{x}_i = 1, b_{ij^*k_i} = 1$ 
     $n \leftarrow n + 1$ 
End While

```

Figure 7.7. Locating the sinks

If every point j is covered and budget constraint is satisfied in period t , we consider the next period $t + 1$. If the coverage of the sensor field is obtained but the budget constraint is violated, unused sensors until period t (i.e. they have been in standby mode until the current period) are removed starting with those having the largest cost c_k . If it is impossible to satisfy the budget constraint even after the removal of some sensors, the algorithm in Figure 7.5 is terminated with $L' = t - 1$. In case a coverage constraint is not satisfied in period t , additional sensors have to be activated. The first step is to compute the undercoverage U_{jt} of each point j in period t as

$$U_{jt} = \max \left(0, d_j - \sum_{i=1}^J \sum_{k=1}^K a_{ijk} z_{ikt} \right). \quad (7.21)$$

There are two possible strategies to ensure the coverage of all points, i.e. to make $U_{jt} = 0$ for all j . These are (i) activating a sensor at point i that is in standby mode in period t , and (ii) placing a new sensor at point i , and activating it. It is reasonable

-
1. While $\{(i, k, t) : x_{ik} = 1, z_{ikt} = 0, E_{ikt}^{\text{rem}} \geq (e_{ik}^s + h_{ik}e_k^c)\} \neq \emptyset$
and $no_con1(i^*, k^*, t) > 0$ Do

$(i^*, k^*) = \arg \max_{\{(i,k): x_{ik}=1, z_{ikt}=0, E_{ikt}^{\text{rem}} \geq (e_{ik}^s + h_{ik}e_k^c)\}} \{no_con1(i, k, t)\}$
 Activate a type- k^* sensor at point i^* , i.e. set $z_{i^*k^*t} = 1$
 If the network is connected Then
 $L'' = t$, and Stop
 Else
 Go to Step 2
 End If
 End While
 2. While $no_con2(i^*, k^*, t)/c_{ik} > 0$ and $\{(i, k) : x_{ik} = 0, c_{ik} \leq B^{\text{rem}}\} \neq \emptyset$ Do

$(i^*, k^*) = \arg \max_{\{(i,k): c_{ik} \leq B^{\text{rem}}\}} \left\{ \frac{no_con2(i, k, t)}{c_{ik}} \right\}$
 Place and activate a type- k^* sensor at point i^*
 If the network is connected Then
 $L'' = t$, and Stop
 End If
 End While
 $L'' = t - 1$
-

Figure 7.8. Restoring feasibility with respect to connectivity

to give priority to the first strategy since the second one reduces the remaining budget by deploying a new sensor. Consequently, among the already deployed sensors, the one having the largest coverage-energy product P_{ikt} is selected for activation. P_{ikt} is defined as

$$P_{ikt} = \left(\sum_{j=1}^J a_{ijk} U_{jt} \right) q_{ikt}, \quad (7.22)$$

where q_{ikt} is the remaining energy of a type- k sensor placed at point i in period t .

The activation of the sensors in standby mode is continued until either a sensor with remaining energy sufficient to be active during a period cannot be found or there is no reduction in the undercoverage U_{jt} . Then, the second strategy is implemented, namely, the best sensor to be deployed is determined. This is done by computing the coverage-cost ratio

$$R_{ikt} = \frac{\left(\sum_{j=1}^J a_{ijk} U_{jt} \right) E_k}{c_{ik}} \quad (7.23)$$

for each type- k sensor that would be placed at point i . The one having the largest ratio R^* is activated, and undercoverage values are updated. The procedure is repeated until complete coverage of the sensor field is obtained for period t . If $R^* = 0$ during the procedure, then it is not possible to satisfy the coverage constraint in period t , and $L' = t - 1$. If the budget constraint does not allow to locate a new sensor, we remove some of the sensors that are not used until period t (i.e. they have been in standby mode until the current period). If the budget constraint is still not satisfied, then we set $L' = t - 1$.

The output of the algorithm in Figure 7.5 is a partial solution in which all points are covered for periods $t = 1, \dots, L'$, and the total cost of deployed sensors is within the budget. This partial solution has to be converted to a complete solution by determining sink locations and data transmission routes from each active sensor to a sink in each period. Therefore, in Step 3 of the algorithm in Figure 7.3 we use the algorithm in Figure 7.7 to locate the sinks. To this end, we first define an indicator variable $\bar{x}_i, i = 1, \dots, J$ where $\bar{x}_i = 1$ if there is a sensor at point i (i.e. $x_{ik} = 1$ for any $k = 1, \dots, K$), and $\bar{x}_i = 0$ otherwise. Then, we rank potential locations $j = 1, \dots, J$ for the sinks with respect to the number of sensors which have point j in the communication range. In other words, for each point j we count the number of sensors that can transmit their data directly to a sink that would be located at this point. In case more than one sensor is placed at point i , we only consider the sensor having the smallest communication range, which is denoted by k_i . The reason is that other sensors with longer transmission ranges can transmit data to the sink if type- k_i can do so. For this

purpose, we use variable $no_cov(j)$. Recall that we set the communication parameter $b_{ijk} = 1$ if point j is within the communication range of a type- k sensor placed at point i ; otherwise $b_{ijk} = 0$. Thus, $no_cov(j) = \sum_{i=1}^J b_{ijk_i} \bar{x}_i$. Point j with the largest $no_cov(j)$ value, j^* , is selected as the first sink location. After the first sink is located, we update the indicator variables $\bar{x}_i = 1$ by setting $\bar{x}_i = 0$ for those points i which can transmit their data to the sink at j^* . The computation of $no_cov(j)$ and the selection of the best point j^* are repeated until S sinks are located. The algorithm in Figure 7.7 locates the sinks in such a way that as many sensors as possible can reach a sink directly, which implies that there may be sensors that cannot send their data directly to a sink. Furthermore, the applied method does not take into account the information provided by the algorithm in Figure 7.5 as to which sensors are active in each period. Therefore, for sensors that cannot transmit their data to a sink directly, it may also be impossible to reach a sink via other sensors. In other words, a connected WSN is not guaranteed in every period $t = 1, \dots, L'$. For this purpose, we run the algorithm in Figure 7.8 in Step 4 of the algorithm in Figure 7.3.

Starting with the first period (i.e. $t = 1$), we check the connectivity of the network in each period. If the network is disconnected in any period t , we apply the following method. First, existing sensors that are in standby mode in period t are determined (i.e. sensors with $x_{ik} = 1$ and $z_{ikt} = 0$). Among them, the sensor which decreases the number of disconnected sensors the most (or increases the number of connected sensors the most) and also has enough energy for the activation in this period is selected and activated. In the algorithm in Figure 7.8, we use parameter $no_con1(i, k, t)$ to denote the additional number of sensors that become connected when an existing type- k sensor at point i is activated in period t . Mathematically, the selected sensor is $(i^*, k^*) = \arg \max_{(i, k): E_{ikt}^{\text{rem}} \geq (e_{ik}^s + h_{ik} e_k^c)} \{no_con1(i, k, t)\}$ where E_{ikt}^{rem} is the remaining energy of a type- k sensor at point i in period t . If the network is still disconnected, this procedure is repeated until either the connectivity is obtained in period t or no existing sensor in standby mode can be activated. Then, we consider the option of locating new sensors as long as their cost does not lead to the violation of the budget constraint. Activating existing sensors takes priority over locating new sensors because the first alternative does not incur an additional cost. We define

parameter $no_con2(i, k, t)$ as the additional number of sensors that become connected when a type- k sensor is located at point i in period t . The best location and type of the new sensor is determined as $(i^*, k^*) = \arg \max_{(i,k): c_{ik} \leq B^{\text{rem}}} \{no_con2(i, k, t)/c_{ik}\}$ where $B^{\text{rem}} = B - \sum_{k=1}^K \sum_{i=1}^J c_{ik} x_{ik}$ is the remaining budget. The rationale behind this criterion is that we want to maximize the number of connected sensors per unit cost. If the network is still disconnected, this procedure is repeated until either the connectivity is obtained in period t or B^{rem} is not sufficient for the placement of a new sensor that can improve the connectivity. In the latter case, $L'' = t - 1$. It is important to note that $L'' \leq L'$ holds. Namely, the network lifetime may be reduced when we try to assure the connectivity in addition to the coverage constraints.

At the end of the algorithm in Figure 7.8, we obtain a partial solution in which coverage constraints are satisfied, a connected WSN is obtained for periods $t = 1, \dots, L''$ and the total cost remains within the budget. This means that in each period $t = 1, \dots, L''$ all the points of the sensor field are covered by sensors and these sensors can transmit their data to a sink either directly or via other sensors. The last step of the algorithm in Figure 7.3, which is Step 5, involves the determination of the routes in which the collected data of the sensors flow to a sink. This is accomplished by solving an LP which has the maximization of L_{LB} as the objective function where L_{LB} , a lower bound on the network lifetime, is less than or equal to L'' which is the objective value of the partial solution at the end of the algorithm in Figure 7.8. The decision variables of this LP are the flow variables f_{ikjlt} and g_{ikjlt} . Its first set of constraints are the flow balance constraints which imply that in each period $t \leq L_{LB}$, the sum of the data generated and the data received from other active sensors should be equal to the sum of the data sent to other active sensors and to the sinks for every active type- k sensor at point i . The second set of constraints ensure that the total energy spent by each deployed sensor for sensing and processing, receiving data, and transmitting data during L_{LB} periods does not exceed the battery energy of the sensor. The final constraints are upper bounds on the flow capacities for sensor-to-sensor and sensor-to-sink flows.

The objective value of this solution is used as $L_{LB}^{(n)}$ in Step 5 of the Lagrangean Heuristic (LH) given as the algorithm in Figure 7.1. Based on Karmarkar's interior-

point algorithm, the LP used to find the optimal data routes is of complexity $O((\tau^{c_1})^3 J^3 K^6 T^3 L_{B_2})$ where L_{B_2} is the number of bits necessary to store the LP instance.

Proposition 22. *The complexity of the proposed LH is polynomial in the problem size and is given as*

$$\begin{aligned} \mathcal{O} \left(\text{iterlim} \left[(\tau^{c_1})^3 J^3 K^6 T^3 L_{B_1} + \tau^{s_1} \tau^{s_2} JKT + \tau^{s_1} J^2 K^2 T + \tau^{c_2} J^2 K^3 T \right. \right. \\ \left. \left. + (\tau^{c_1})^3 J^3 K^6 T^4 L_{B_2} \right] \right) \end{aligned} \quad (7.24)$$

where $\tau^{s_2} = \max_j \left\{ \sum_{i=1}^J \sum_{k=1}^K a_{ijk} \right\}$ is the maximum number of sensors covering a point of the sensor field.

Proof. We repeat *iterlim* times the following operations that contribute significantly to the complexity. We solve the third subproblem ($(\tau^{c_1})^3 J^3 K^6 T^3 L_{B_1}$ term of the expression). We restore coverage and budget constraints feasibility ($\tau^{s_1} \tau^{s_2} JKT + \tau^{s_1} J^2 K^2 T$ term of the expression). We restore feasibility with respect to connectivity ($\tau^{c_2} J^2 K^3 T$ term of the expression). We solve the LP used to find the optimal data routes ($(\tau^{c_1})^3 J^3 K^6 T^4 L_{B_2}$ term of the expression). Q.E.D.

8. A COLUMN GENERATION BASED HEURISTIC

8.1. Introduction

In this chapter⁴ we propose a Column Generation Based Heuristic (CGH) for LSRP. In CGH we first reformulate LSRP. We solve the LP relaxation of the reformulation using the column generation method. Each new column corresponds to a sensor set that satisfies the coverage and flow balance constraints. In the final step of CGH we construct a feasible solution to LSRP using the sensor sets obtained during the column generation phase.

8.2. Determination of Sink Locations

The first step of CGH involves the determination of the sink locations, which helps in reducing the number of variables in the formulation of LSRP. An iterative procedure is used to locate the sinks where the potential locations for sinks $j = 1, \dots, J$ are ranked on the basis of the number of sensors that have point j in their communication range. In other words, for each point j in the sensor field the number of sensors that can transmit their data directly to a sink that would be located at this point is determined. After a sink is placed at the winner point, the procedure is repeated with the only difference that sensors which can transmit their data directly to the previously located sink are not considered. Figure 8.1 outlines the procedure used for locating the sinks, where $\bar{x}_{ik}$ for $i = 1, \dots, J$, $k = 1, \dots, K$ is an indicator variable which is equal to one if sensor (i, k) cannot transmit its data to any sink located earlier, and zero otherwise. $\tau(j)$ denotes the number of sensors that cannot transmit their data directly to a sink placed earlier in the procedure, but can transmit their data directly to the sink that would be located at point j . At the end of the algorithm, $s_j^* = 1$ if a sink is located at point j , and $s_j^* = 0$ otherwise. After the determination of the sink locations, it is still difficult to obtain a solution of the LP relaxation of LSRP due to the large number of variables

⁴The paper Türkoğulları et al. [70] including the results of this chapter is accepted for publication in the European Journal of Operational Research.

-
1. Initialization: Set $\bar{x}_{ik} = 1$ for $i = 1, \dots, J$, $k = 1, \dots, K$, and $n = 1$
 2. While $n \leq S$ Do

Compute $\tau(j) = \sum_{i=1}^J \sum_{k=1}^K b_{ijk} \bar{x}_{ik}$ for $j = 1, \dots, J$
 Locate a sink at $j^* = \arg \max_j \{\tau(j)\}$
 $\bar{x}(i, k) = 0$ for i, k such that $\bar{x}_{ik} = 1$ and $b_{ij^*k} = 1$
 $n \leftarrow n + 1$
- End While
-

Figure 8.1. Algorithm location of sinks

and constraints. Therefore, we opt for applying column generation (Desaulniers et al. [71]) to solve the LP relaxation of a reformulation of the original problem, which makes up the second step of CGH. The idea of Dantzig and Wolfe [72], namely, working only with a subset of the variables and extending a linear program column-wise as needed in the solution process is a viable strategy to solve large-scale linear programs.

8.3. Reformulation

Let vector $\mathbf{z}^\alpha$ represent an active sensor subset in any time period t whose elements are sufficient to cover every point in the sensor field with the required quality. Such a sensor subset can be described in the following way: its component $z_{ik}^\alpha = 1$ if sensor (i, k) is in the active subset, $z_{ik}^\alpha = 0$ otherwise. In fact, a sequence $\mathbf{z}^1, \dots, \mathbf{z}^L$ corresponds to any feasible solution of LSRP with lifetime value L , i.e. in any time period t within the lifetime L , there is an active sensor subset that can be represented by a vector $\mathbf{z}^\alpha$. In the sequel, a sequence $\mathbf{z}^1, \dots, \mathbf{z}^L$ is referred to as a sensor schedule. We note that the set of different sensor schedules is a finite set, because the total number of sensors that can be located in the field is limited. In this section, we reformulate LSRP using active sensor subsets represented by $\mathbf{z}^\alpha$ that form the sensor schedule $\mathbf{z}^1, \dots, \mathbf{z}^L$. For this reformulation, we also need to define data flow routes that are used to transmit

the data collected by the elements of the sensor subset $\mathbf{z}^\alpha$ to the sinks. Let f_{ikjl}^α and g_{ikj}^α denote, respectively, the feasible flow (i.e. feasible with respect to flow balance and the capacity restrictions) from sensor (i, k) to sensor (j, l) , and the feasible flow from sensor (i, k) to the sink at j , whose location is determined in the first step of CGH. For the sake of notational clarity, we let $\mathbf{f}^\alpha$ and $\mathbf{g}^\alpha$ be the vectors whose components are f_{ikjl}^α and g_{ikj}^α respectively. Formally, $(\mathbf{z}^\alpha, \mathbf{f}^\alpha, \mathbf{g}^\alpha)$ satisfy constraints (8.1) - (8.6) given as follows:

$$\sum_{i=1}^J \sum_{k=1}^K a_{ijk} z_{ik}^\alpha \geq d_j \quad \text{for all } j \quad (8.1)$$

$$h_{ik} z_{ik}^\alpha + \sum_{\substack{j=1 \\ j \neq i}}^J \sum_{l=1}^K b_{jil} f_{jlik}^\alpha = \sum_{\substack{j=1 \\ j \neq i}}^J \sum_{l=1}^K b_{ijk} f_{ikjl}^\alpha + \sum_{\substack{j=1 \\ s_j^*=1}}^J b_{ijk} g_{ikj}^\alpha \quad \text{for all } i, k \quad (8.2)$$

$$f_{ikjl}^\alpha \leq u_{kl} b_{ijk} z_{ik}^\alpha \quad \text{for all } i, k, j \neq i, l \quad (8.3)$$

$$f_{ikjl}^\alpha \leq u_{kl} b_{ijk} z_{jl}^\alpha \quad \text{for all } i, k, j \neq i, l \quad (8.4)$$

$$g_{ikj}^\alpha \leq v_k b_{ijk} z_{ik}^\alpha \quad \text{for all } i, k, \{j : s_j^* = 1\} \quad (8.5)$$

$$z_{ik}^\alpha \in \{0, 1\}, f_{ikjl}^\alpha \geq 0, g_{ikj}^\alpha \geq 0. \quad (8.6)$$

Here parameters $J, K, a_{ijk}, b_{ijk}, d_j, h_{ik}, u_{kl}$, and v_k are as defined in Chapter 5. We realize that f_{ikjl}^α is positive only if sensor (i, k) and sensor (j, l) are active in period t . Similarly, g_{ikj}^α is positive only if sensor (i, k) is active in period t . As will be seen later, the reformulation of LSRP has columns for each $(\mathbf{z}^\alpha, \mathbf{f}^\alpha, \mathbf{g}^\alpha)$ satisfying (8.1) - (8.6). We note that, for a given active sensor subset $\mathbf{z}^\alpha$, there can be infinitely many flow vectors $\mathbf{f}^\alpha$ and $\mathbf{g}^\alpha$ such that $(\mathbf{z}^\alpha, \mathbf{f}^\alpha, \mathbf{g}^\alpha)$ satisfy (8.1) - (8.6). Consequently, if we do not put any restriction on $\mathbf{f}^\alpha$ and $\mathbf{g}^\alpha$ for a given $\mathbf{z}^\alpha$, our reformulation becomes a semi-infinite program. In order to obtain a finite integer program whose LP relaxation can be solved by standard column generation, we restrict the set $\Psi = \{\alpha : (\mathbf{z}^\alpha, \mathbf{f}^\alpha, \mathbf{g}^\alpha) \text{ satisfies (8.1) - (8.6)}\}$. We only consider those sensor subsets that make sensor schedules feasible with respect to the energy constraints (5.6) or that correspond to extreme points of the convex hull of the mixed-integer set defined by (8.1) - (8.6). This restriction is on the basis of the following fact. We can in principle find an optimal solution to LSRP by considering only the subset of sensor schedules in which flow vectors $\mathbf{f}^\alpha$ and $\mathbf{g}^\alpha$ are assigned to sensor subsets $\mathbf{z}^\alpha$ in all time periods $t \leq L$ in such a way that $(\mathbf{z}^\alpha, \mathbf{f}^\alpha, \mathbf{g}^\alpha)$ satisfy

(8.1) - (8.6) and the energy constraints (5.6). Formally, this subset of sensor schedules is $\Theta = \{\mathbf{z}^1, \dots, \mathbf{z}^L : \text{there exists } (\mathbf{z}^1, \mathbf{f}^1, \mathbf{g}^1), \dots, (\mathbf{z}^L, \mathbf{f}^L, \mathbf{g}^L) \text{ satisfying (8.1) - (8.6) with } \sum_{t=1}^L \gamma_{ik}^t \leq E_k \text{ for all sensors } (i, k)\}$. Here, γ_{ik}^α is the energy consumption of sensor (i, k) in sensor subset $\mathbf{z}^\alpha$ while accomplishing its sensing, processing, data receiving and transmission tasks. Mathematically, it is expressed as

$$\gamma_{ik}^\alpha = e_{ik}^s z_{ik}^\alpha + e_k^r \sum_{\substack{j=1 \\ j \neq i}}^J \sum_{l=1}^K b_{jil} f_{jlik}^\alpha + \sum_{\substack{j=1 \\ j \neq i}}^J e_k^c \sum_{l=1}^K b_{ijk} f_{ikjl}^\alpha + \sum_{\substack{j=1 \\ s_j^*=1}}^J e_k^c b_{ijk} g_{ikj}^\alpha. \quad (8.7)$$

To each schedule $\mathbf{z}^1, \dots, \mathbf{z}^L$ of Θ , we assign one set of flow vectors $(\mathbf{z}^1, \mathbf{f}^1, \mathbf{g}^1), \dots, (\mathbf{z}^L, \mathbf{f}^L, \mathbf{g}^L)$ satisfying (8.1) - (8.6) with $\sum_{t=1}^L \gamma_{ik}^t \leq E_k$ for all sensors (i, k) . Among all schedules of Θ whose active sensor subsets satisfy the budget constraint (5.8), the one with the highest lifetime value L becomes the activity schedule of an optimal solution to LSRP. Therefore, in our reformulation, among $(\mathbf{z}^\alpha, \mathbf{f}^\alpha, \mathbf{g}^\alpha) \in \Psi$ we consider those that define sensor activity schedules in Θ . Formally, we consider $\{(\mathbf{z}^\alpha, \mathbf{f}^\alpha, \mathbf{g}^\alpha) : \alpha \in M^\Theta\}$ where $M^\Theta = \{\alpha : (\mathbf{z}^\alpha, \mathbf{f}^\alpha, \mathbf{g}^\alpha) \text{ is the sensor subset and the corresponding flow vectors in a time period } t \leq L \text{ in at least one sensor schedule } \mathbf{z}^1, \dots, \mathbf{z}^L \text{ in } \Theta\}$. Since the set of different sensor schedules is finite and we assign only one set of triplets $(\mathbf{z}^1, \mathbf{f}^1, \mathbf{g}^1), \dots, (\mathbf{z}^L, \mathbf{f}^L, \mathbf{g}^L)$ to each element $\mathbf{z}^1, \dots, \mathbf{z}^L$ of Θ , M^Θ is a finite set. We define another finite index set M^E that consists of $(\mathbf{z}^\alpha, \mathbf{f}^\alpha, \mathbf{g}^\alpha)$ triplets which correspond to an extreme point of the convex hull of the mixed-integer set defined by (8.1) - (8.6). Formally, $M^E = \{\alpha : (\mathbf{z}^\alpha, \mathbf{f}^\alpha, \mathbf{g}^\alpha) \text{ is an extreme point of the convex hull of the mixed-integer set defined by (8.1) - (8.6)}\}$. In solving the LP relaxation of the reformulation using column generation, after finding an initial set of columns corresponding to $\alpha \in M^\Theta$, considering only new columns corresponding to $\alpha \in M^E$ is sufficient to obtain an optimal solution. Consequently, $(\mathbf{z}^\alpha, \mathbf{f}^\alpha, \mathbf{g}^\alpha)$ triplets from $\{(\mathbf{z}^\alpha, \mathbf{f}^\alpha, \mathbf{g}^\alpha) : \alpha \in M^\Theta \cup M^E\}$ is used in the reformulation. In the sequel, the union of the sets M^Θ and M^E is denoted by M , i.e. $M = M^\Theta \cup M^E$. We note that not only the sensor subset $\mathbf{z}^\alpha$, but also flow vectors $\mathbf{f}^\alpha$ and $\mathbf{g}^\alpha$ assigned to this sensor subset characterize elements of the index set M . For example, α_1 and α_2 with $\mathbf{z}^{\alpha_1} = \mathbf{z}^{\alpha_2}$ can both be in M if $\mathbf{f}^{\alpha_1} \neq \mathbf{f}^{\alpha_2}$ or $\mathbf{g}^{\alpha_1} \neq \mathbf{g}^{\alpha_2}$ since in a sensor schedule $\mathbf{z}^1, \dots, \mathbf{z}^L$ in Θ the same sensor subset may be active in two different time periods with different sensor-to-sink

data flow routes.

Now, LSRP can be reformulated as the following “Reformulated LSRP” (LSRP”) using the fixed sink locations $\mathbf{s}^*$.

LSRP”:

$$\max \sum_{t=1}^T \sum_{\alpha \in M} \lambda_{t\alpha} \quad (8.8)$$

s.t.

$$\sum_{\alpha \in M} \lambda_{t\alpha} \leq 1 \quad \text{for all } t \quad (8.9)$$

$$\sum_{t=1}^T \sum_{\alpha \in M} \gamma_{ik}^\alpha \lambda_{t\alpha} \leq E_k \quad \text{for all } (i, k) : z_{ik}^\alpha = 1 \text{ for at least one } \alpha \in M \quad (8.10)$$

$$\sum_{\alpha \in M} z_{ik}^\alpha \lambda_{t\alpha} - x_{ik} \leq 0 \quad \text{for all } (i, k, t) : z_{ik}^\alpha = 1 \text{ for at least one } \alpha \in M \quad (8.11)$$

$$\sum_{k=1}^K \sum_{i=1}^J c_{ik} x_{ik} \leq B \quad (8.12)$$

$$\sum_{i=1}^J \sum_{k=1}^K c_{ik} x_{ik} \geq \sigma \quad (8.13)$$

$$\sum_{\alpha \in M} \lambda_{(t+1)\alpha} \varphi^\alpha - \sum_{\alpha \in M} \lambda_{t\alpha} \varphi^\alpha \leq 0 \quad t = 1, \dots, T-1 \quad (8.14)$$

$$x_{ik} \in \{0, 1\}, \lambda_{t\alpha} \in \{0, 1\} \quad (8.15)$$

Observe that the variables of LSRP” are x_{ik} and $\lambda_{t\alpha}$. As defined before, $x_{ik} = 1$ if a type- k sensor is placed at point i of the sensor field, $x_{ik} = 0$ otherwise. $\lambda_{t\alpha} = 1$ if sensor subset $\mathbf{z}^\alpha$ is active in period t , $\lambda_{t\alpha} = 0$ otherwise. z_{ik}^α is a parameter in this formulation since sensor (i, k) ’s membership status in sensor subset $\mathbf{z}^\alpha$ is known. Objective function (8.8) maximizes the network lifetime expressed as the total active time of all sensor subsets in $\{\mathbf{z}^\alpha : \alpha \in M\}$ over all periods $t = 1, \dots, T$. Hence, it is equivalent to the objective function of LSRP. Constraints (8.9) ensure that at most one sensor subset is active in any period t . Constraints (8.10) guarantee that the total energy consumed by sensor (i, k) during active times of sensor subsets in $\{\mathbf{z}^\alpha : \alpha \in M\}$ to which it belongs is less than the initial battery energy. Constraints (8.11) enforce that sensor subset $\mathbf{z}^\alpha$ including sensor (i, k) can be active in period t only if sensor

(i, k) is placed. Constraint (8.12) makes sure that the total cost of deployed sensors does not exceed the budget. In fact, constraints (8.10), (8.11), and (8.12) correspond to constraints (5.6), (5.7), and (5.8) of LSRP, respectively. Constraint (8.13) is a valid inequality. In (8.13), σ is a lower bound on the minimum cost of a sensor subset covering every point of the sensor field with the required coverage quality. It can be obtained by solving the LP relaxation of an integer program with the objective of minimizing the cost of the deployed sensors. This integer program should contain constraints similar to (8.1) to ensure the coverage of the sensor field by the deployed sensors at the required quality level. We use constraint (8.13) to strengthen the formulation of RP. Because every sensor subset in $\{\mathbf{z}^\alpha : \alpha \in M\}$ satisfies the coverage constraint by definition (see inequality (8.1)), Constraint (8.13) is valid for the convex hull of the mixed-integer set defined by LSRP".

Assuming that $\mathbf{z}^1, \dots, \mathbf{z}^L$ is an optimal sensor schedule, any permutation of $\mathbf{z}^1, \dots, \mathbf{z}^L$ is also optimal. Constraints (8.14) are introduced to break this symmetry. They state that active sensor subset in period $t + 1$ is required to consume less energy than the active sensor subset in period t . In constraints (8.14), φ^α is the total energy consumption of the sensor subset $\mathbf{z}^\alpha$, which is defined as

$$\varphi^\alpha = \sum_{i=1}^J \sum_{k=1}^K \gamma_{ik}^\alpha. \quad (8.16)$$

We want to point out that reformulations based on active sensor subsets are also used in Cardei et al. [17] and Alfieri et al. [19] for their problems which focus on one of the four issues considered in the integrated problem LSRP. It must be also emphasized that constraints of LSRP related to the coverage requirement constraints (5.3), data flow balance constraints (5.5), and flow capacity constraints (5.10) - (5.13) are satisfied automatically by variables $\mathbf{z}^\alpha$, $\mathbf{f}^\alpha$, and $\mathbf{g}^\alpha$ due to the definition of the index set M .

8.4. Solving the LP Relaxation of the Reformulation Using Column Generation

Although M is a finite set, the number of the $\lambda_{t\alpha}$ variables in LSRP'' is huge due to the large number of elements in M . Hence, the column generation (CG) algorithm is suitable to solve the LP relaxation of LSRP'' referred to as LP-LSRP''. To speed up CG, an initial index set $M^{(0)} \subset M$ consisting of $\alpha \in M$ obtained from the sensor schedule of a good feasible solution to LSRP is formed. For this purpose, connected sensor subsets are found, which are disjoint from each other, and cover the sensor field with the required quality when their sensors are all active. A linear program is solved to determine the data flow routes from sensors to sinks that maximize the lifetime of each disjoint subset, when the sensors in the considered subset are active, and sensors in other subsets are in standby mode. This implies that in the feasible solution thus obtained, only elements of one disjoint sensor subset is active in any given period, and as soon as one of the sensors in this subset runs out of energy, all sensors are switched to standby mode, and another sensor subset is activated. This procedure will be explained in detail in Chapter 9. Throughout the CG algorithm, indices $\alpha \in M \setminus M^{(0)}$ are added to $M^{(0)}$, until an optimal solution of LP-LSRP'' is obtained. The restricted master problem LP-LSRP'' formed with $\alpha \in M^{(n)}$ at iteration n is called LP-LSRP'' _{n} , and given as follows:

LP-LSRP''_n:

$$\max \sum_{t=1}^T \sum_{\alpha \in M^{(n)}} \lambda_{t\alpha} \quad (8.17)$$

s.t.

$$\text{Constraints (8.9) with } M \text{ replaced by } M^{(n)} \quad (8.18)$$

$$\text{Constraints (8.10) with } M \text{ replaced by } M^{(n)} \quad (8.19)$$

$$\text{Constraints (8.11) with } M \text{ replaced by } M^{(n)} \quad (8.20)$$

$$\text{Constraints (8.12)–(8.13)} \quad (8.21)$$

$$\text{Constraints (8.14) with } M \text{ replaced by } M^{(n)} \quad (8.22)$$

$$x_{ik} \leq 1 \quad (8.23)$$

$$x_{ik} \geq 0, \lambda_{t\alpha} \geq 0. \quad (8.24)$$

Note that due to the dependence of constraints (8.19) and (8.20) on α , the addition of the variable $\lambda_{t\alpha}$, and the corresponding column to LP-LSRP''_n may require further constraints to be added to LP-LSRP''_n. Note that columns corresponding to the variables with negative (positive) reduced cost which are found by the pricing subproblems are added to a minimization-type (maximization-type) restricted master problem in the classical CG algorithm, but the constraint set of the restricted master problem remains always the same until an optimal solution is obtained for the master problem. This is not the case in LP-LSRP''. Therefore, standard optimality conditions of LP theory (Bazaraa et al. [73]) are not trivially valid for this problem. However, due to the specific structure of the formulation that includes the constraints related to $\alpha \in M^{(n)}$, an optimality condition can still be obtained. In other words, it is possible to obtain a termination criterion for the solution of LP-LSRP'' based on the solution of LP-LSRP''_n. The following proposition provides this criterion.

Proposition 23. *Let w_t^{1*} , w_{ik}^{2*} , w_{ikt}^{3*} , w_t^{4*} be the optimal values of the dual variables of LP-LSRP''_n corresponding to constraints (8.18), (8.19), (8.20), and (8.22) respectively. If these values satisfy the following constraints, then the optimal solution of LP-LSRP''_n*

is sufficient to determine an optimal solution of LP-LSRP".

$$w_t^1 + \sum_{\substack{i=1 \\ (i,k) \in \theta^{(n)}}}^J \sum_{k=1}^K \gamma_{ik}^\alpha w_{ik}^2 + \sum_{\substack{i=1 \\ (i,k) \in \theta^{(n)}}}^J \sum_{k=1}^K z_{ik}^\alpha w_{ikt}^3 - \varphi^\alpha w_t^4 + \varphi^\alpha w_{t-1}^4 \geq 1 \quad t = 1, \dots, T; \alpha \in M^E \quad (8.25)$$

where $\theta^{(n)} = \{(i, k) : z_{ik}^\alpha = 1 \text{ for at least one } \alpha \in M^{(n)}\}$.

Proof. To show this, we first define a problem called $\overline{\text{LP-LSRP}}_n''$, using variables $\{x_{ik}, i = 1, \dots, J; k = 1, \dots, K\}$ and $\{\lambda_{t\alpha}, t = 1, \dots, T; \alpha \in M\}$. The constraints of $\overline{\text{LP-LSRP}}_n''$ are exactly of the same type as those of $\text{LP-LSRP}_n''$, but written with a larger set of $\lambda_{t\alpha}$ variables. This means that the only difference between the two mathematical programs is the number of $\lambda_{t\alpha}$ variables. The $\lambda_{t\alpha}$ variables of $\overline{\text{LP-LSRP}}_n''$ are defined for $t = 1, \dots, T$ and $\alpha \in M$, whereas those of $\text{LP-LSRP}_n''$ for $t = 1, \dots, T$ and $\alpha \in M^{(n)}$. We also note the relation between $\overline{\text{LP-LSRP}}_n''$ and the LP-LSRP". Both problems have the same variables, but $\overline{\text{LP-LSRP}}_n''$ has only a subset of the constraints of LP-LSRP", i.e. constraints (8.10) and (8.11) corresponding to $\alpha \in M^{(n)}$.

Let us consider the dual problems of $\text{LP-LSRP}_n''$ and $\overline{\text{LP-LSRP}}_n''$, with dual variables $w_t^1, w_{ik}^2, w_{ikt}^3, w_t^4$ corresponding to constraints (8.18), (8.19), (8.20), and (8.22) respectively. Observe that these are the only constraints containing the primal variables $\lambda_{t\alpha}$. Since the right-hand sides of the constraints of the primal problems are the same, the objective functions of both programs are also the same. The difference between the dual problems occurs in the constraints corresponding to the $\lambda_{t\alpha}$ variables. These constraints are given as

$$w_t^1 + \sum_{\substack{i=1 \\ (i,k) \in \theta^{(n)}}}^J \sum_{k=1}^K \gamma_{ik}^\alpha w_{ik}^2 + \sum_{\substack{i=1 \\ (i,k) \in \theta^{(n)}}}^J \sum_{k=1}^K z_{ik}^\alpha w_{ikt}^3 - \varphi^\alpha w_t^4 + \varphi^\alpha w_{t-1}^4 \geq 1 \quad t = 1, \dots, T; \alpha \in M^{(n)} \quad (8.26)$$

for the dual problem of $\text{LP-LSRP}_n''$, while the same constraints are written for the dual problem of $\overline{\text{LP-LSRP}}_n''$ using all $\alpha \in M$. Now, let $w_t^{1*}, w_{ik}^{2*}, w_{ikt}^{3*}, w_t^{4*}$ be the optimal values of the dual variables of $\text{LP-LSRP}_n''$ corresponding to constraints (8.18), (8.19), (8.20), and (8.22) respectively. If these values also satisfy the additional constraints of the dual of $\overline{\text{LP-LSRP}}_n''$ written for $\alpha \in M \setminus M^{(n)}$, as implied by satisfaction of constraints

(8.25), then $w_t^{1*}, w_{ik}^{2*}, w_{ikt}^{3*}, w_t^{4*}$ are also feasible and optimal values for the dual problem of $\overline{\text{LP-LSRP}}_n''$. By the strong duality theorem of LP, the optimal objective values of the primal problems $\text{LP-LSRP}_n''$ and $\overline{\text{LP-LSRP}}_n''$ must be the same as well. Let $(\mathbf{x}^*, \lambda^*)$ denote an optimal solution of $\text{LP-LSRP}_n''$, and construct the following feasible solution $(\mathbf{x}^{**}, \lambda^{**})$ to $\overline{\text{LP-LSRP}}_n''$ from $(\mathbf{x}^*, \lambda^*)$,

$$\begin{cases} x_{ik}^{**} = x_{ik}^* & \text{for all } i, k \\ \lambda_{t\alpha}^{**} = \lambda_{t\alpha}^* & \text{for all } t \text{ and } \alpha \in M^{(n)} \\ \lambda_{t\alpha}^{**} = 0 & \text{for all } t \text{ and } \alpha \in M \setminus M^{(n)}. \end{cases} \quad (8.27)$$

Since the objective values of $\text{LP-LSRP}_n''$ for $(\mathbf{x}^*, \lambda^*)$ and $\overline{\text{LP-LSRP}}_n''$ for $(\mathbf{x}^{**}, \lambda^{**})$ are the same (i.e., $\sum_{t=1}^T \sum_{\alpha \in M^{(n)}} \lambda_{t\alpha}^* = \sum_{t=1}^T \sum_{\alpha \in M} \lambda_{t\alpha}^{**}$), and $(\mathbf{x}^*, \lambda^*)$ is an optimal solution of $\text{LP-LSRP}_n''$, it follows that $(\mathbf{x}^{**}, \lambda^{**})$ is an optimal solution of $\overline{\text{LP-LSRP}}_n''$. Recall the relation between $\overline{\text{LP-LSRP}}_n''$ and $\text{LP-LSRP}''$. Since both problems have the same variables and $\overline{\text{LP-LSRP}}_n''$ has only a subset of the constraints of $\text{LP-LSRP}''$, it is easy to conclude that the optimal value of $\overline{\text{LP-LSRP}}_n''$ is an upper bound on the optimal value of $\text{LP-LSRP}''$. We can observe that $(\mathbf{x}^{**}, \lambda^{**})$ is a feasible solution to $\text{LP-LSRP}''$, because its additional constraints, i.e. constraints (8.10) and (8.11) for $\alpha \in M \setminus M^{(n)}$ are trivially satisfied by $(\mathbf{x}^{**}, \lambda^{**})$ due to the fact that $\lambda_{t\alpha}^{**} = 0$ for all t and $\alpha \in M \setminus M^{(n)}$. Moreover, the objective value of $\text{LP-LSRP}''$ corresponding to $(\mathbf{x}^{**}, \lambda^{**})$ is equal to the upper bound given by the optimal value of $\overline{\text{LP-LSRP}}_n''$, and $(\mathbf{x}^{**}, \lambda^{**})$ is therefore optimal. Q.E.D.

If at the end of iteration n of the CG algorithm constraints (8.25) are satisfied, $\text{LP-LSRP}''$ is solved to optimality by Proposition 23, and we can execute the third step of the CGH heuristic, where a feasible solution to LSRP is constructed from the solution of $\text{LP-LSRP}''$ in which decision variables x_{ik} and $\lambda_{t\alpha}$ may have fractional values. Otherwise, that is, if some of the constraints (8.25) are not satisfied, then the most violated constraint corresponding to $\alpha^* \in M^E \setminus M^{(n)}$ is added to the dual problem of $\text{LP-LSRP}_n''$, which corresponds to adding columns $\{\lambda_{t\alpha^*}, t = 1, \dots, T\}$ to $\text{LP-LSRP}_n''$. α^* can efficiently be found by solving the following pricing subproblem in time period t , called PS_t , using a commercial software such as CPLEX [62].

PS_t :

$$\begin{aligned} \max \quad & 1 - w_t^{1*} - \sum_{i=1}^J \sum_{k=1}^K \gamma_{ik}^\alpha w_{ik}^{2*} - \sum_{i=1}^J \sum_{k=1}^K z_{ik}^\alpha w_{ikt}^{3*} + \varphi^\alpha w_t^{4*} - \varphi^\alpha w_{t-1}^{4*} \\ \text{s.t.} \quad & (8.1)-(8.6). \end{aligned} \quad (8.28)$$

After finding α^* , the procedure is repeated by solving LP-LSRP''_{n+1} with $M^{(n+1)} = M^{(n)} \cup \alpha^*$. Since M^E is a finite set, second step of CGH terminates in a finite number of iterations. Second step of CGH is summarized in Figure 8.2.

-
1. Initialization: Form the initial index set $M^{(0)}$ as described at the beginning of Section 3.3. Set $n = 0$
 2. While constraints (8.25) are not satisfied Do
 - Solve LP-LSRP''_n
 - Solve PS_t for $t = 1, \dots, T$
 - If constraints (8.25) are satisfied Then
 - Solve LP-LSRP'' to optimality using Proposition 23
 - Else
 - $M^{(n+1)} = M^{(n)} \cup \alpha^*$
 - $n \leftarrow n + 1$
 - End If
 - End While
-

Figure 8.2. Solving the LP relaxation of RP by column generation

8.5. Construction of a Feasible Solution

As pointed out, the third step of CGH constructs a feasible solution to LSRP. To this end, we use another heuristic called CFSL which is used for *Constructing a Feasible Solution to LSRP*. In CFSL sensor subsets obtained during CG are used. At

each iteration of CFSL an eligible sensor subset is activated for one period. During this period only sensors of the selected subset are set to active mode while other sensors are kept in standby mode. An eligible sensor subset is the one whose additional cost of activation is less than the remaining budget, and whose remaining lifetime is greater than one period. Sensor subset to be activated is selected as follows. Among the sensor subsets that have additional cost of activation the one having the highest lifetime-to-cost ratio is given priority. Among the sensor subsets that do not have any additional cost the one with the highest remaining lifetime is chosen. Final selection between the two prospective sensor subsets of both groups is made based on remaining lifetime. CFSL forms a sensor schedule by selecting sensor subsets and by activating them one period. Sensor schedule formation terminate when all the sensor subsets obtained during CG become ineligible. A formal description of CFSL is given in the following. Let N denote the total number of iterations performed in finding an optimal solution of LP-LSRP" using CG. CFSL uses variables x_{ik}^f and λ_α^f to designate, respectively, a feasible solution for sensor placement and the total active time of sensor subset $\mathbf{z}^\alpha$ with the initial values $\mathbf{x}^f = \lambda^f = \mathbf{0}$. Note that $x_{ik}^f = 1$ if a sensor of type- k is placed at point i , $x_{ik}^f = 0$ otherwise. At each iteration, CFSL selects a sensor subset among eligible sensor subsets defined as $\{\mathbf{z}^\alpha : \alpha \in M^{(N)} \setminus \Phi\}$ to be activated, where Φ denotes the set of ineligible sensor subsets identified at previous iterations with $\Phi = \emptyset$ initially. Activating the sensor subset $\mathbf{z}^\alpha$ necessitates the placement of its sensors, i.e. setting $x_{ik}^f = 1$ for sensors with $z_{ik}^\alpha = 1$. There are two criteria that determine the eligibility of a sensor subset. First, the additional cost of activating a sensor subset given as $\bar{c}_\alpha = \sum_{i=1}^J \sum_{k=1}^K c_{ik} (1 - x_{ik}^f) z_{ik}^\alpha$ must be less than the remaining budget $B^r = B - \sum_{i=1}^J \sum_{k=1}^K c_{ik} x_{ik}^f$. Second, the energy of some of the sensors in a subset might already be consumed by other previously activated subsets containing these sensors such that this sensor subset has a remaining lifetime less than one period. Such a sensor subset has to be treated as ineligible. Since the remaining lifetime of a sensor subset $\mathbf{z}^\alpha$ is equal to the minimum remaining lifetime of the sensor in the subset, it can be mathematically defined as $L_\alpha^r = \min_{(i,k): z_{ik}^\alpha = 1} \{E_{ik}^r / \gamma_{ik}^\alpha\}$ where $E_{ik}^r = \max \{E_k - \sum_{\alpha \in M_N} \lambda_\alpha^f \gamma_{ik}^\alpha, 0\}$ is the remaining energy of sensor (i, k) , and γ_{ik}^α is the energy consumption of sensor (i, k) in sensor subset $\mathbf{z}^\alpha$ defined in expression (8.7).

One of the tasks to be performed in an iteration is to find the remaining lifetime L_α^r and the additional cost $\bar{c}_\alpha$ of sensor subsets in $\{\mathbf{z}^\alpha : \alpha \in M^{(N)} \setminus \Phi\}$. The other is to label the sensor subsets with $\bar{c}_\alpha > B^r$ or $L_\alpha^r < 1$ as ineligible and add them to Φ . Considering only the eligible sensor subsets in $\{\mathbf{z}^\alpha : \alpha \in M^{(N)} \setminus \Phi\}$, the sensor subset $\mathbf{z}^{\tilde{\alpha}}$ to be activated at the current iteration is selected as follows. First, all sensor subsets are partitioned into two groups depending on whether their additional cost $\bar{c}_\alpha$ of activation is positive or zero. Among those with $\bar{c}_\alpha > 0$, the one with the highest lifetime-to-cost ratio $q_\alpha = L_\alpha^r / \bar{c}_\alpha$ is the prospective sensor subset for activation. If there is a tie with respect to q_α , the sensor subset with the highest remaining lifetime L_α^r is given priority to keep the energy level of the sensor with the minimum remaining energy as high as possible. If two or more sensor subsets have the same L_α^r value, the one with the highest total remaining energy $\sum_{i=1}^J \sum_{k=1}^K E_{ik}^r z_{ik}^\alpha$ is selected. The prospective sensor subset from the other group of sensor subsets with $\bar{c}_\alpha = 0$ is selected as the one with the highest value of L_α^r . If there is a tie, the one having the highest total remaining energy is chosen from this group. The final selection for the sensor subset $\mathbf{z}^{\tilde{\alpha}}$ to be activated is made from the two prospective sensor subsets of both groups. The sensor subset with higher remaining lifetime value L_α^r is activated for one period. λ_α^f , the total active time of sensor subset $\mathbf{z}^{\tilde{\alpha}}$, and the final lifetime value L^f found by CFSL are increased by one period. During this period, only sensors of subset $\mathbf{z}^{\tilde{\alpha}}$ are set to active mode, while other sensors are kept in standby mode. Data collected by the sensors of subset $\mathbf{z}^{\tilde{\alpha}}$ are transmitted to the sinks using the routes indicated by $\mathbf{f}^{\tilde{\alpha}}$ and $\mathbf{g}^{\tilde{\alpha}}$, which were found while generating the sensor subset in CG.

This procedure is repeated until no eligible sensor subset remains to be activated, i.e. $\Phi = M^{(N)}$. This occurs when either the remaining budget B^r is not sufficient for the activation of a new sensor subset, or all sensor subsets have a remaining lifetime $L_\alpha^r < 1$ due to insufficient remaining energy of the sensors. The CFSL heuristic is summarized in Figure 8.3.

-
1. Initialization: Set $\mathbf{x}^f = 0$, $\lambda^f = 0$, $\Phi = \emptyset$, $E_{ik}^r = E_k$, $L^f = 0$, $B^r = B$
 2. While $M^{(N)} \setminus \Phi \neq \emptyset$ Do
 3. For all sensor subsets in $\{\mathbf{z}^\alpha : \alpha \in M^{(N)} \setminus \Phi\}$ Do
 - Calculate L_α^r and $\bar{c}_\alpha$
 - If $L_\alpha^r < 1$ or $\bar{c}_\alpha > B^r$ Then $\Phi \leftarrow \Phi \cup \alpha$
 - End For
 4. Partition the sensor subsets in $\{\mathbf{z}^\alpha : \alpha \in M^{(N)} \setminus \Phi\}$ into two groups depending on whether $\bar{c}_\alpha > 0$ or $\bar{c}_\alpha = 0$
 5. Determine prospective sensor subsets for activation from both groups on a set of criteria. Select subset $\mathbf{z}^{\tilde{\alpha}}$ to be activated from the two prospective sensor subsets
 6. Set $\lambda_{\tilde{\alpha}}^f \leftarrow \lambda_{\tilde{\alpha}}^f + 1$ and $L^f \leftarrow L^f + 1$
 Update $\mathbf{x}^f, B^r, \mathbf{E}^r$
 - End While
 7. Return $\mathbf{x}^f, L^f, \lambda^f$
-

Figure 8.3. Algorithm construction of a feasible solution to LSRP

8.6. Implementation Issues

The proposed heuristic CGH starts with the determination of sink locations. In the second step, LSRP is reformulated using active sensor subsets and the determined sink locations. The LP relaxation of the reformulation, LP-LSRP", is solved by CG. Finally, in the third step, a feasible solution to LSRP is constructed from the optimal solution of LP-LSRP" using a heuristic called CFSL.

We would like to mention a few implementation issues about the deployment scheme of the proposed sensor network. It can be used in applications such as environmental monitoring and precision agriculture. Since the decisions about the sensor and sink locations are made in a centralized manner using the integrated model, the deployment of sensors and sinks can be realized by using vehicles or manually depend-

ing on the size of the sensor field. The proposed scheme also includes the optimal activity schedule of the sensors to provide an extended lifetime, and this schedule has to be preloaded to the sensors before the deployment. This long-term scheduling requires time synchronization among the sensors, which can be achieved by either using GPS or proxy synchronization. Since the length of the time periods (also called time buckets) are selected as 12 hours in our computational study, which is relatively long, a coarse/rough synchronization is sufficient. More details about synchronization in WSNs can be found in Sundararaman et al. [74]. The joint optimization scheme developed in this study also prescribes the optimal routes employed in each time bucket. These routes can be implemented for each sensor using next hop routing by table lookup in each period where the tables are preloaded to each sensor before the deployment. To avoid storage problems in sensors, the sink can be utilized to upload the necessary routing tables to the sensors every two or three months. More detailed information about centralized routing can be found in Stathopoulos et al. [75].

9. DISJOINT SETS HEURISTIC

9.1. Introduction

In this chapter⁵ we present a fast and efficient heuristic for LSRP. The heuristic called *Disjoint Sets Heuristic* (DSH) first finds the locations of the sinks using a simple algorithm. It then determines a sensor set disjoint from the previously selected sets, that covers the sensor field with the required quality, and that has minimum cost. The next step is to make the sensor set connected by enlarging it if necessary. DSH then determines sensor-to-sink data flow routes that maximize the set's lifetime. In its final step, DSH removes cycles from the flow quantities. In each time period only one disjoint sensor set is active, others are in standby mode in order to save energy.

9.2. Disjoint Sets Heuristic

At each main iteration $n = 1, \dots, N$ of DSH, where N is the total number of iterations, a subset Ψ_n of the sensors is found. It is disjoint from the previously selected subsets, which is ensured in the first step by defining a set Φ including all the sensors that belong to previously selected sets. Initially, $\Phi = \emptyset$. Also, L_{DSH} , the lifetime value calculated by DSH, is initialized to zero. The second step is the location of the sinks. In order to reduce the largest distance from a sensor to a sink, we evenly locate the sinks by dividing the two-dimensional (three-dimensional) sensor field into S rectangular (prismatic) parts of equal area (volume) and locating each sink at the center of each part. As the largest distance decreases, the maximum energy expenditure in data transmission becomes smaller, because it is a function of the distance. The third step is the selection of a minimum cost sensor set Ψ_n disjoint from Φ which covers all points j of the field with the required quality d_j . In small-sized problem instances, an integer programming model called *Minimum Cost Coverage* (MCC) is solved for this purpose.

⁵The paper Türkoğulları et al. [76] including the results of this chapter appeared in Ad Hoc Networks

MCC:

$$\min \sum_{\substack{i=1 \\ (i,k) \notin \Phi}}^I \sum_{k=1}^K c_{ik} w_{ik} \quad (9.1)$$

s.t.

$$\sum_{\substack{i=1 \\ (i,k) \notin \Phi}}^I \sum_{k=1}^K a_{ijk} w_{ik} \geq d_j \quad j = 1, \dots, J \quad (9.2)$$

$$w_{ik} \in \{0, 1\} \quad i = 1, \dots, I; k = 1, \dots, K; (i, k) \notin \Phi. \quad (9.3)$$

In MCC, parameters c_{ik} , a_{ijk} , d_j , I , J , and K are the same as in LSRP presented in Chapter 5. w_{ik} is a variable taking value one if sensor (i, k) is in the set Ψ_n , and zero otherwise. The objective function (9.1) minimizes the cost of Ψ_n . Constraints (9.2) ensure that every point j of the sensor field is covered with the required quality d_j . Notice that only $(i, k) \notin \Phi$ is eligible to be in Ψ_n .

Since MCC contains the NP-complete set covering problem as a special case and is difficult to solve exactly, we devise a heuristic called *Disjoint Set Finder* (DSF) in order to solve large problem instances. Starting with an empty set Ψ_n , DSF adds sensors to Ψ_n one by one until all points are covered with d_j sensors. Sensors (i, k) that are not in Φ and cover at least one point j in the field with insufficient coverage are eligible to be added to Ψ_n . Among the eligible sensors, the one having the smallest cost-to-coverage ratio

$$\varphi_{ik} = \frac{c_{ik}}{\sum_{j=1}^J a_{ijk} \alpha_j} \quad (9.4)$$

is given priority, where α_j is the shortage in the coverage of point j given by

$$\alpha_j = \max \left(0, d_j - \sum_{\substack{i=1 \\ (i,k) \in \Psi_n}}^I \sum_{k=1}^K a_{ijk} \right). \quad (9.5)$$

DSF is summarized in Figure 9.1.

Proposition 24. *The complexity of DSF is $\mathcal{O}(\tau^s J^2 K^2)$, where $\tau^s = \max_{i,k} \left\{ \sum_{j=1}^J a_{ijk} \right\}$ is the maximum number of points in the sensor field that a sensor can cover.*

Proof. In the worst case we repeat JK times the following operations. We calculate all φ_{ik} values ($\mathcal{O}(\tau^s JK)$). We find the maximum argument ($\mathcal{O}(JK)$). We update α_j ($\mathcal{O}(\tau^s)$). Q.E.D.

DSF is an adaptation of the recent greedy approximation algorithm proposed by Altinel et al. [12] to solve the differentiated coverage problem in heterogeneous sensor networks. Also, the greedy measure (9.4) becomes the one used by Cardei et al. [17] for $c_{ik} = 1$. A connected sensor set is the one in which all sensors can transmit the

1. Initialization $\Psi_n \leftarrow \emptyset, \alpha_j \leftarrow d_j$ for all j
 2. While $\alpha_j > 0$ for some $j \in \{1, \dots, J\}$ Do
 - If there is a sensor $(i, k) \notin \Phi$ with $\sum_j a_{ijk} \alpha_j > 0$ Then
 - Determine sensor (i^*, k^*) with the smallest φ_{ik} value
 - $\Psi_n \leftarrow \Psi_n \cup (i^*, k^*)$ and $\Phi \leftarrow \Phi \cup (i^*, k^*)$
 - Update α_j for all j
 - Else /* there is no sensor $(i, k) \notin \Phi$ with $\sum_j a_{ijk} \alpha_j > 0$ */
 - A feasible sensor set Ψ_n cannot be found, stop
 - End If
 - End While
 3. Return Ψ_n
-

Figure 9.1. Algorithm disjoint set finder

data they collect to a sink either directly or through neighboring sensors. As shown by Zhang and Hou [63], coverage implies connectivity in a homogeneous WSN (i.e. a network with a single type of sensor) if the communication range of the sensor is more than twice of its sensing range. Unfortunately, this is not always true in heterogeneous WSNs. A pathological example is given in Appendix A. Consequently, in the fourth

step of DSH we try to make sure that sensor set Ψ_n is connected. This is achieved by a procedure called *Enlargement of the Disjoint Set for Connectivity* (EDSC) which works on a set Ψ_n^d consisting of the disconnected sensors in Ψ_n . EDSC adds sensors to Ψ_n one by one until all disconnected sensors can transmit their data to a sink through the newly added sensors. Obviously, a sensor eligible for being added to Ψ_n must not be in Φ , and its cost c_{ik} must be less than the remaining budget $B^r = B - \sum_{i=1}^J \sum_{k=1}^K c_{ik}$. Among all eligible sensors, the one having the largest connectivity-to-cost ratio q_{ik} is given priority. It is defined as

$$q_{ik} = \frac{\rho_{ik}}{c_{ik}}, \quad (9.6)$$

where ρ_{ik} is the number of sensors in Ψ_n^d which become connected with the addition of sensor (i, k) to Ψ_n . EDSC is outlined in Figure 9.2.

Proposition 25. *The complexity of EDSC is $\mathcal{O}(\tau^{c_1} J^3 K^3)$, where $\tau^{c_1} = \max_j \left\{ \sum_{i=1}^J \sum_{k=1}^K b_{ijk} \right\}$ is the maximum number of sensors having a point of the sensor field within their transmission range.*

Proof. Initialization is $\mathcal{O}(JK)$. Finding connected sensors in Ψ_n is $\mathcal{O}(\tau^{c_1} JK)$. Calculating all q_{ik} is $\mathcal{O}(\tau^{c_1} J^2 K^2)$. Updating Ψ_n^d is $\mathcal{O}(\tau^{c_1} JK)$. We repeat JK times in the worst case. Q.E.D.

At the end of the fourth step of DSH, disjoint set Ψ_n does not only cover all points of the sensor field with the required quality, but also forms a connected WSN. The fifth step of DSH is to find L_{Ψ_n} , i.e. the duration of time in which disjoint set Ψ_n is active. Notice that during L_{Ψ_n} only the sensors in Ψ_n are active, while others are in standby mode to save energy. Since the final lifetime L_{DSH} is the sum of all active times L_{Ψ_n} corresponding to sets Ψ_n , $n = 1, \dots, N$, L_{Ψ_n} should be maximized in DSH. The battery energies associated with different sensor types, E_k , $k = 1, \dots, K$, limit the

-
1. Initialization $\Psi_n^d \leftarrow \Psi_n$
 2. Find connected sensors in Ψ_n and remove them from Ψ_n^d
 3. While $(\Psi_n^d \neq \emptyset)$ Do
 - If there is a sensor $(i, k) \notin \Phi$ with $c_{ik} \leq B^r$ Then
 - Determine sensor (i^*, k^*) with the highest q_{ik} value
 - $\Psi_n \leftarrow \Psi_n \cup (i^*, k^*)$ and $\Phi \leftarrow \Phi \cup (i^*, k^*)$
 - Update Ψ_n^d by removing sensors that become connected due to addition of (i^*, k^*) to Ψ_n
 - Else /* there is no sensor $(i, k) \notin \Phi$ with $c_{ik} \leq B^r$ */
 - A connected set Ψ_n cannot be obtained, stop
 - End If
 - End While
 4. Return Ψ_n
-

Figure 9.2. Algorithm enlargement of the disjoint set for connectivity

active time L_{Ψ_n} of sensor set Ψ_n . This relationship can be expressed as

$$L_{\Psi_n} = \min_{(i,k) \in \Psi_n} \frac{E_k}{\gamma_{ik}} \quad (9.7)$$

where γ_{ik} is the energy consumption rate of sensor $(i, k) \in \Psi_n$ while accomplishing its sensing, data processing, data receiving and transmitting tasks. These tasks have to be performed collectively by each sensor (i, k) in Ψ_n to cover all the points of the sensor field with the required quality and to transmit the collected data to sinks. γ_{ik} is formally given as

$$\gamma_{ik} = e_k^s + e_k^r \sum_{\substack{j=1 \\ j \neq i, (j,l) \in \Psi_n}}^I \sum_{l=1}^K b_{jl} \bar{f}_{jlik} + \sum_{j=1}^I e_k^c \sum_{\substack{l=1 \\ l \neq i, (j,l) \in \Psi_n}}^K b_{ijk} \bar{f}_{ikjl} + \sum_{\substack{j=1 \\ \bar{s}_j=1}}^I e_k^c b_{ijk} \bar{g}_{ikj}. \quad (9.8)$$

In (9.8), e_k^s, e_k^r, e_k^c and b_{ijk} are communication coefficients that are defined in Chapter 5. $\bar{f}_{ikjl}$ is the amount of data packets transmitted from sensor (i, k) to sensor (j, l) in

a period. $\bar{g}_{ikj}$ is the amount of data packets transmitted from sensor (i, k) to a sink at point j in a period. $\bar{s}_j$ is a parameter which is equal to one if a sink is located at point j , and zero otherwise. We notice that γ_{ik} is a linear function of variables $\bar{\mathbf{f}}$ and $\bar{\mathbf{g}}$. Consequently, optimal values $\bar{\mathbf{f}}^*$ and $\bar{\mathbf{g}}^*$ of variables $\bar{\mathbf{f}}$ and $\bar{\mathbf{g}}$ can be found by solving the following linear program (LPFG) to maximize L_{Ψ_n} defined in (9.7).

LPFG:

$$\max \quad \delta \tag{9.9}$$

s.t.

$$\delta \leq \frac{E_k}{\gamma_{ik}} \quad (i, k) \in \Psi_n \tag{9.10}$$

$$h_{ik} + \sum_{j=1}^I \sum_{l=1}^K b_{jil} \bar{f}_{jlik} = \sum_{j=1}^I \sum_{l=1}^K b_{ijk} \bar{f}_{ikjl} + \sum_{j=1}^I b_{ijk} \bar{g}_{ikj} \quad (i, k) \in \Psi_n \tag{9.11}$$

$$\bar{\mathbf{f}}, \bar{\mathbf{g}}, \delta \geq 0 \tag{9.12}$$

The objective function (9.9) maximizes the active time $\delta = \min_{(i,k) \in \Psi_n} (E_k/\gamma_{ik})$ of sensor set Ψ_n . Constraints (9.10) state that δ is determined by the shortest active time of all the sensors in Ψ_n . Constraints (9.11) are the flow balance equations ensuring that the sum of the total data inflow to sensor (i, k) and the data it generates is equal to the total data outflow from that sensor. As in Chapter 5, h_{ik} is the number of data packets generated by a type- k sensor at point i during a period. The maximum active time $L_{\Psi_n}^*$ of sensor set Ψ_n is then given as δ^* , the optimal value of δ .

Proposition 26. *Based on Karmarkar's interior point algorithm [60] the complexity of LPFG is $\mathcal{O}((\tau^{c_2})^3 J^3 K^6 \theta)$, where $\tau^{c_2} = \max_{i,k} \left\{ \sum_{j=1}^J b_{ijk} \right\}$ is the maximum number of points of the sensor field within the transmission range of a sensor, and θ is the number of binary bits required to store a LPFG instance.*

Proof. Number of variables is $\mathcal{O}(\tau^{c_2} JK^2)$. Number of constraints is $\mathcal{O}(JK)$. Q.E.D.

In the optimal sensor-to-sensor flow values $\bar{\mathbf{f}}^*$, some data packets transmitted

from sensor (i, k) of sensor set Ψ_n may return to itself if the lifetime of sensor (i, k) given by E_k/γ_{ik} and lifetimes of other sensors (j, l) transmitting these packets given by E_l/γ_{jl} are higher than the active time δ^* of sensor set Ψ_n . This characterizes a cycle. To eliminate such cycles a procedure called *Removing Cycles* (RC) is applied. In the first step of RC, auxiliary variables $\tilde{\mathbf{f}}$ and $\tilde{\mathbf{g}}$ are initialized to $\tilde{\mathbf{f}}^*$ and $\tilde{\mathbf{g}}^*$, while variables $\mathbf{f}_{DSH}$ and $\mathbf{g}_{DSH}$ denoting the final flow quantities with no cycles are set to $\mathbf{0}$. At each main iteration, a path p with no cycles is built from sensor (i, k) to a sink, and some of the data generated by sensor (i, k) is transmitted to that sink along p . In other words, path p is a finite sequence of sensors in Ψ_n starting with sensor (i, k) , and its last element is a sink. There is a positive flow as an entry of $\tilde{\mathbf{f}}$ between two successive elements of path p if those two elements are both sensors and there is a positive flow as an entry of $\tilde{\mathbf{g}}$ if one of them is a sink. We call these two successive elements an *edge* of p . Positive flow on an edge is called *edge flow*.

We use variable λ_{ik} , which is initialized to h_{ik} , to keep track of sensor (i, k) 's remaining data to be routed to a sink. In building a path p for each sensor (i, k) with positive λ_{ik} , all sensor-to-sink flow values $\tilde{g}_{ikj}$ are examined first. If there is a positive flow to a sink at point j , path p ends with an edge from sensor (i, k) to the sink at j . If there is more than one positive flow to sinks, the one with the larger flow quantity is chosen. If there is no positive flow to a sink, flows $\tilde{f}_{ikjl}$ to other sensors (j, l) are examined. Sensor (j^*, l^*) being the receiver of the largest outflow from sensor (i, k) constitutes the next edge of path p with (i, k) . Notice that a positive flow either to a sink or to a neighboring sensor should exist due to flow balance constraints (9.11). This procedure is repeated for sensor (j^*, l^*) . Sensors are added to p unless a sink is the last element. During sensor additions to p , sensor (j, l) that is already an element of an existing edge may be selected as the new element of the path. This means that part of p starting and ending with sensor (j, l) is a cycle. In that case, it is removed from p , and $\tilde{\mathbf{f}}$ is updated to remove the cycle from its elements. To this end, the smallest edge flow quantity σ^l among the edges of the cycle is found and subtracted from the corresponding components of $\tilde{\mathbf{f}}$. Path formation of p is restarted from the element before sensor (j, l) . After building a path p from sensor (i, k) to a sink, smallest edge flow quantity σ^p among the edges of the path is calculated and $\min(\sigma^p, \lambda_{ik})$ is

transmitted to a sink on path p . Consequently, an amount of data equal to $\min(\sigma^p, \lambda_{ik})$ is added to the components of $\mathbf{f}_{DSH}$ and $\mathbf{g}_{DSH}$ corresponding to edge flows on path p , whereas an amount of $\min(\sigma^p, \lambda_{ik})$ is subtracted from the corresponding components of $\tilde{\mathbf{f}}$ and $\tilde{\mathbf{g}}$, because these flow quantities are dedicated to the transmission of the data generated by sensor (i, k) . λ_{ik} is also reduced by an amount of $\min(\sigma^p, \lambda_{ik})$. Paths that transmit the data of sensor (i, k) to sinks are continued to be built until λ_{ik} becomes zero in the second step of RC. RC is summarized in Figure 9.3. Notice that at each run of the “Do while” loop in the second step of RC either λ_{ik} or a component of $\tilde{\mathbf{f}}$ or $\tilde{\mathbf{g}}$ becomes zero. This ensures that RC terminates in a finite number of iterations.

Proposition 27. *The complexity of RC is in the order $\mathcal{O}((\tau^{c_2})^2 J^2 K^4)$.*

Proof. Initialization is $\mathcal{O}(\tau^{c_2} JK^2)$. Forming a path from sensor (i, k) to a sink without a loop is $\mathcal{O}(\tau^{c_2} JK^2)$. Loop removal is $\mathcal{O}(JK)$. At each iteration either a component of $\tilde{\mathbf{f}}$ is made zero or a λ_{ik} is made zero. Therefore at most $\mathcal{O}(\tau^{c_2} JK^2)$ iterations. Q.E.D.

At the end of the fifth step of DSH, a sensor set Ψ_n covering all the points of the grid with the required quality as well as optimal paths without any cycles transmitting sensors' collected data to sinks are found. Sensors being elements of Ψ_n are active for $L_{\Psi_n}^*$ periods and transmit data to sinks using the optimal paths found. Other sensors are in standby mode to save energy. At the sixth step of the algorithm, the final network lifetime value, L_{DSH} , is increased by $L_{\Psi_n}^*$ periods, and the elements of Ψ_n are included in Φ , the set of previously used sensors. Steps 1–6 of DSH are repeated until a connected set Ψ_n covering the points of the sensor field with the required quality disjoint from Φ cannot be found. Since at least one sensor is necessary to satisfy the coverage requirement, the total number of disjoint sets, N , is less than or equal to number JK of available sensors. DSH is outlined in Figure 9.4.

Proposition 28. *The complexity of DSH is*

$$\mathcal{O}(\tau^s J^3 K^3 + \tau^{c_1} J^4 K^4 + (\tau^{c_2})^3 J^4 K^7 \theta + (\tau^{c_2})^2 J^3 K^5). \quad (9.13)$$

-
1. Initialization: Set $\tilde{\mathbf{f}} = \bar{\mathbf{f}}^*$, $\tilde{\mathbf{g}} = \bar{\mathbf{g}}^*$, $\lambda_{ik} = h_{ik}$, $\mathbf{f}_{DSH} = \mathbf{0}$, $\mathbf{g}_{DSH} = \mathbf{0}$
 2. For all sensors $(i, k) \in \Psi_n$ Do
 - While $\lambda_{ik} > 0$ Do
 - Form a path p from sensor (i, k) to a sink using $\tilde{\mathbf{f}}, \tilde{\mathbf{g}}$
 - If a cycle is detected Then
 - subtract σ^l from components of $\tilde{\mathbf{f}}$ corresponding to edge flows of the cycle
 - Else /* a path p without a cycle is formed */
 - Add an amount of $\min(\sigma^p, \lambda_{ik})$ packets to each component of $\mathbf{f}_{DSH}$ and $\mathbf{g}_{DSH}$ corresponding to edge flows of p
 - Subtract an amount of $\min(\sigma^p, \lambda_{ik})$ packets from components of $\tilde{\mathbf{f}}$ and $\tilde{\mathbf{g}}$ corresponding to edge flows of p
 - $\lambda_{ik} \leftarrow \lambda_{ik} - \min(\sigma^p, \lambda_{ik})$
 - End If
 - End While
 - End For
 3. Return $\mathbf{f}_{DSH}, \mathbf{g}_{DSH}$
-

Figure 9.3. Algorithm removing cycles

Proof. There can be at most JK disjoint sets because there are at most JK sensors. The above expression is obtained after summing the complexity terms of each of the four steps of DSH and multiplying with JK . Q.E.D.

-
1. Initialization: Set $L_{DSH} = 0, \Phi = \emptyset, n = 1$
 2. Divide the two (three) dimensional sensor field into S rectangular (prismatic) parts of equal area (volume) and locate each sink at the center of each part
While (a connected set Ψ_n satisfying coverage requirement is found) Do
 3. For small size problems solve integer program MCC, for large size problems use the algorithm in Figure 9.1 to obtain a sensor set Ψ_n satisfying the coverage requirement.
If a set Ψ_n satisfying coverage requirement disjoint from Φ is found Then
 4. Use the algorithm in Figure 9.2 to make Ψ_n connected
If the algorithm in Figure 9.2 can satisfy connectivity Then
 5. Solve LPFG to find optimal data transmission paths from sensors to sinks. Remove cycles in the optimal flow vector $\bar{\mathbf{f}}^*$ using the algorithm in Figure 9.3
 6. $L_{DSH} \leftarrow L_{DSH} + L_{\Psi_n}^*, \Phi \leftarrow \Phi \cup \Psi_n, n \leftarrow n + 1$
End If
 - End If
 - End While
-

Figure 9.4. Algorithm disjoint sets heuristic

10. COMPUTATIONAL STUDY

10.1. Introduction

In this chapter we assess experimentally the performance of the methods proposed in the thesis. We first find the accuracy of the solution method proposed to solve LSP. We then assess the strength of the valid inequality and the cutting plane proposed in Chapter 4. Next, we test the strength of the valid inequalities proposed for LSRP. We then evaluate the quality of the upper bound found by the branch and price algorithm. Next, we find the accuracies of the heuristic methods proposed for LSRP. We then investigate the efficiency of the DSH in solving very large LSRP instances. Finally we use the proposed methods to reveal some new aspects of the LSRP.

All the experiments are carried out on a computer with Intel Xeon 3.16 GHz processor and 28 GB of RAM working under Windows 2003 Server operating system. In the experiments we assume that the sensor field has a rectangular grid structure with $m_1 \times m_2$ points and the coverage requirements are set to $d_j = 2$ for all points in the sensor field. There are $K = 2$ sensor types, and parameter values that depend on the sensor type are given in Table 10.1. The value of e_k^s is based on a period length of 12 hours. It is assumed that a data packet is generated every half an hour, i.e. 1800 seconds. This implies that data generation rate per period, h_k , is equal to 24 data packets. The values of e_k^s, e_k^r, e_k^c , and E_k are based on the experimental results for a Mica2 mote provided by Torres and Kabara [77]. Upper bounds on data flows (u_{kl} and v_k) are selected as large as 1×10^8 data packets for all $k = 1, 2$ and $l = 1, 2$. The number of sinks to be located is $S = 3$.

Table 10.1. Sensor specifications

k	c_k	r_k^s	r_k^c	e_k^s (Joules)	e_k^r (Joules)	e_k^c (Joules)	E_k (Joules)
1	1	1	2	744	0.01	0.013	86,400
2	2	2	4	744	0.01	0.018	86,400

In the experiments, for each network instance we assign three values to B . Low

budget level corresponds to the case where only 75% of the points can be deployed with a type-1 sensor, while all the points of the sensor field can be deployed with type-2 sensors for the case of high budget level. Medium budget level is the average of the low and high budget levels.

10.2. Location and Scheduling Problem

In this section we determine the accuracy and efficiency of the Lagrangean heuristic (LH) proposed in Chapter 3. We compare the results provided by LH with those given by the solver CPLEX 11.0 [62]. Table 10.2 shows the results. For each problem instance we first use LH to obtain a feasible solution and an upper bound on the optimal value. We denote by L_{LB}^* the lifetime value of the feasible solution given by LH which is a lower bound on the optimal objective. We indicate by L_{UB}^* the upper bound on the optimal objective found by LH. We then use CPLEX 11.0 with the default options. In each problem instance we impose a time limit which is larger than the CPU time of LH to CPLEX 11.0. If CPLEX 11.0 finds an optimal solution the lifetime value of the feasible solution L_{LB}^C and the upper bound on the optimal objective L_{UB}^C are the same. Also the CPU time is less than the imposed time limit. In some large-sized problem instances CPLEX 11.0 cannot provide a feasible solution within the imposed time limit. These cases are indicated by “-” in the table. We see that LH finds a better solution than CPLEX 11.0 in 12 of the 21 instances. CPLEX 11.0 finds a better solution in 5 instances. They find the same feasible solution in one instance. CPLEX 11.0 cannot find a feasible solution in the remaining 3 instances. When we compare the upper bounds found we see that LH gives a better upper bound in 7 of the 21 instances. CPLEX 11.0 finds a better upper bound in 11 instances. They find the same upper bound in one instance. CPLEX 11.0 cannot find an upper bound in the remaining 2 instances.

We express the accuracy of the methods as percent deviation from the best known

upper bound. For instance the accuracy of LH is

$$100 \times \frac{\min(L_{UB}^*, L_{UB}^C) - L_{LB}^*}{\min(L_{UB}^*, L_{UB}^C)}. \quad (10.1)$$

Based on this formula the average accuracy of LH is 30.8%. The average accuracy of CPLEX 11.0 is 45.7%. LH is more accurate than CPLEX 11.0. When we look at the average CPU times we see that the average CPU time consumed by LH is 513.6 seconds, whereas the average CPU time consumed by CPLEX 11.0 is 3298.3 seconds. LH is more efficient than CPLEX 11.0.

In order to see the performance of LH in large LSP instances we perform further experiments. We vary $J = m_1 \times m_2$ from 80 to 400. The results are shown in Table 10.3. In these large instances, although we give CPLEX 11.0 more time than the CPU time consumed by LH, CPLEX 11.0 provides neither a feasible solution, nor an upper bound. In these large instances, the accuracy of LH is 42.3. LH spends 7055.3 seconds of CPU time on the average.

10.3. Covering Integer Program

To assess the usefulness of the proposed valid inequality (4.14) and the cutting plane (4.64) we worked with the test problems drawn from Beasley's OR Library [78]. These are classical set covering problems with $a_{ij} = 0$ or 1 and $b_i = 1$ in (4.1). By using each one of these test problems we generate two new test instances. First of all we keep the same cost vector $\mathbf{c}$ and do not modify the dimensions of matrix A . Then, the first instance is generated by simply replacing the entries that are 1 with random numbers uniformly distributed between 1 and 100 and keeping the zero entries; this gives a sparse test problem. Then we take somewhat the complement of this matrix by replacing its zero entries with the new ones generated using a uniform distribution within $[1, 100]$; this results in a dense instance. In summary, starting with a test problem in [78], we obtained two test problems with dense and sparse A matrices having elements as nonnegative real numbers, while keeping the matrix structure of these well known problems. We have a test bed consisting of 88 test problems. We

Table 10.2. Comparison of LH with CPLEX 11.0

m_1	m_2	B	Network Lifetime				Accuracy		CPU Time (s)	
			L_{LB}^*	L_{UB}^*	L_{LB}^C	L_{UB}^C	LH	$CPLEX11.0$	LH	$CPLEX11.0$
5	2	8	58	79	58	72.5	20	20	63	3600
5	2	14	116	135	106	130.5	11.1	18.8	123	3600
5	2	20	168.5	174	174	174	3.2	0	147	297
5	3	12	58	91.5	87	87	33.3	0	109	568
5	3	21	130	154.5	88.5	151.5	14.2	41.6	169	3600
5	3	30	172	216.5	173.5	206	16.5	15.8	200	3600
5	4	15	58	99.5	57.5	89.5	35.2	35.8	133	3600
5	4	28	116	172	106	164	29.3	35.4	423	3600
5	4	40	185	228.5	188.5	222	16.7	15.1	271	3600
5	5	19	58	111	58.5	97	40.2	39.7	482	3600
5	5	35	116	176	99.5	203.5	34.1	43.5	271	3600
5	5	50	174	242.5	149	247.5	28.2	38.6	296	3600
6	5	23	61	104	53	137.5	41.3	49.0	418	3600
6	5	42	116	189	11.5	223	38.6	93.9	415	3600
6	5	60	174	256	47.5	249	30.1	80.9	405	3600
10	4	30	58	99.5	3	154	41.7	97.0	531	3600
10	4	55	116	181	-	229	35.9	-	835	3600
10	4	80	174	254.5	3	249	30.1	98.8	883	3600
10	6	45	58	168.5	-	-	65.6	-	1082	3600
10	6	83	117	202.5	-	-	42.2	-	1496	3600
10	6	120	176	290.5	3	341.5	39.4	99.0	2033	3600

first solve LPC for each one of them and record the optimal value z^{lpc} and optimal solution x^{lpc} . We then use the algorithm in Figure 4.1 with

$$T_1 = 1, T_2 = 1, M_1 = M, N_1 = N_r \subseteq \left\{ i : x_i^{lpc} > 0 \right\} \quad (10.2)$$

Table 10.3. Performance of LH on large LSP instances

m_1	m_2	B	Network Lifetime		Accuracy	CPU Time (s)
			L_{LB}^*	L_{UB}^*	LH	LH
10	8	60	58	112.5	48.4	2037
10	8	110	116.5	205	43.2	2277
10	8	160	174	264	34.1	2703
10	10	75	58	116	50	3631
10	10	138	123	237.5	48.2	2753
10	10	200	185.5	268	30.8	3506
20	10	150	58	120	51.7	7202
20	10	275	128	240	46.7	5438
20	10	400	190.5	270	29.4	7123
20	15	225	60	123	51.2	11233
20	15	413	130	242	46.3	7852
20	15	600	192	272	29.4	10432
20	20	300	63	125	49.6	14403
20	20	550	133	245	45.7	10988
20	20	800	195	275	29.1	14252

to obtain a valid inequality (4.14). The reason for this choice of parameters (10.2) in applying the algorithm in Figure 4.1 is to select an inequality that cuts off the non integral extreme point x^{lpc} giving z^{lpc} from the class that (4.14) defines. When N_r is chosen as in (10.2), the components of x^{lpc} corresponding to x_{j_l} $l = 1, 2, \dots, |N_3|$ in (4.14) are zero; therefore the inequality does not become weaker in cutting off x^{lpc} as new terms are added during lifting. We then employ the algorithm in Figure 4.2 to obtain a cutting plane of type (4.64). We add the valid inequality (4.14) and the cutting plane (4.64) to the constraint set of LPC (4.5), (4.6) and solve again the problem. We record the new optimal value as z_v^{lpc} .

The results are reported in Table 10.4 and Table 10.5 where the column “instances” shows the names of the test instances drawn from [78]. For each newly gen-

Table 10.4. Optimum values in the first set of test problems

Instances	m	n	Dense			Sparse		
			z^{lpc}	z_v^{lpc}	Cut off x^{lpc}	z^{lpc}	z_v^{lpc}	Cut off x^{lpc}
SCP41	200	1000	1.953	2	Yes (2.41)	484.866	486.776	Yes (0.39)
SCP42	200	1000	2.466	3	Yes (21.65)	534.616	537.566	Yes (0.55)
SCP43	200	1000	2.398	3	Yes (25.10)	527.009	528.923	Yes (0.36)
SCP44	200	1000	1.953	2	Yes (2.41)	502.300	504.478	Yes (0.43)
SCP45	200	1000	2.824	3	Yes (6.23)	560.646	564.595	Yes (0.70)
SCP46	200	1000	2.344	3	Yes (27.99)	609.531	611.522	Yes (0.33)
SCP47	200	1000	2.250	3	Yes (33.33)	466.788	468.053	Yes (0.27)
SCP48	200	1000	2.561	3	Yes (17.14)	509.403	511.465	Yes (0.40)
SCP49	200	1000	2.327	3	Yes (28.92)	669.226	672.075	Yes (0.43)
SCP410	200	1000	2.551	3	Yes (17.60)	582.388	584.533	Yes (0.37)
SCP51	200	2000	2.014	3	Yes (48.90)	271.114	273.008	Yes (0.70)
SCP52	200	2000	2.004	3	Yes (49.70)	307.385	310.748	Yes (1.09)
SCP53	200	2000	1.991	2	Yes (0.45)	243.427	245.252	Yes (0.75)
SCP54	200	2000	2.006	3	Yes (49.55)	247.694	251.032	Yes (1.35)
SCP55	200	2000	1.987	2	Yes (0.65)	233.153	234.032	Yes (0.38)
SCP56	200	2000	1.962	2	Yes (1.94)	234.612	235.798	Yes (0.51)
SCP57	200	2000	1.952	2	Yes (2.46)	306.989	308.401	Yes (0.46)
SCP58	200	2000	1.874	2	Yes (6.72)	289.976	291.337	Yes (0.47)
SCP59	200	2000	2.210	3	Yes (35.75)	301.296	302.763	Yes (0.49)
SCP510	200	2000	2.096	3	Yes (43.13)	273.024	274.812	Yes (0.66)
SCP61	200	1000	2.296	3	Yes (30.66)	134.038	134.803	Yes (0.57)
SCP62	200	1000	2.384	3	Yes (25.84)	140.163	140.855	Yes (0.49)
SCP63	200	1000	2.508	3	Yes (19.62)	149.288	150.603	Yes (0.88)
SCP64	200	1000	1.960	2	Yes (2.04)	138.648	139.313	Yes (0.48)
SCP65	200	1000	2.760	3	Yes (8.70)	158.220	158.860	Yes (0.40)

Table 10.5. Optimum values in the second set of test problems

Instances	m	n	Dense			Sparse		
			z^{lpc}	z_v^{lpc}	Cut off x^{lpc}	z^{lpc}	z_v^{lpc}	Cut off x^{lpc}
SCPA1	300	3000	1.883	2	Yes (6.21)	251.077	251.975	Yes (0.36)
SCPA2	300	3000	2.046	3	Yes (46.63)	254.644	255.316	Yes (0.26)
SCPA3	300	3000	2.008	3	Yes (49.40)	244.767	245.735	Yes (0.40)
SCPA4	300	3000	1.940	2	Yes (3.09)	248.779	249.885	Yes (0.44)
SCPA5	300	3000	1.907	2	Yes (4.88)	243.322	244.394	Yes (0.44)
SCPB1	300	3000	2.009	3	Yes (49.33)	70.857	71.217	Yes (0.51)
SCPB2	300	3000	1.893	2	Yes (5.65)	70.967	71.516	Yes (0.77)
SCPB3	300	3000	2.057	3	Yes (45.84)	77.580	77.927	Yes (0.45)
SCPB4	300	3000	1.942	2	Yes (2.99)	77.914	78.734	Yes (1.05)
SCPB5	300	3000	1.998	2	Yes (0.10)	71.294	71.783	Yes (0.69)
SCPE1	50	500	1.299	2	Yes (53.96)	3.800	4	Yes (5.26)
SCPE2	50	500	1.364	2	Yes (46.63)	3.743	4	Yes (6.87)
SCPE3	50	500	1.216	2	Yes (64.47)	3.479	4	Yes (14.98)
SCPE4	50	500	1.268	2	Yes (57.73)	4.051	5	Yes (23.43)
SCPE5	50	500	1.292	2	Yes (54.80)	3.635	4	Yes (10.04)
SCPC1R10	511	210	1.754	2	Yes (14.03)	24.217	25	Yes (3.23)
SCPC1R11	1023	330	1.801	2	Yes (11.05)	24.515	25	Yes (1.98)
SCPC1R12	2047	495	1.840	2	Yes (8.70)	24.054	25	Yes (3.93)
SCPC1R13	4095	715	1.877	2	Yes (6.55)	23.787	24	Yes (0.90)

erated test problem, the cell under the column “cut off x^{lpc} ” has a “yes” if the added inequalities cut off x^{lpc} and a “no” if this is not the case. In this column in parantheses, improvement in the lower bound given by LPC due to the new valid inequality in percentage of z^{lpc} value is given. Tables 10.4 and 10.5 reveal the fact that in all of 88 problem instances added valid inequality (4.14) and the added cut (4.64) cut off LP relaxation solution x^{lpc} and provide a 12.85% improvement in the lower bound obtained on average.

The execution times are listed in Table 10.6 and Table 10.7 with t denoting the computation time for solving the original LP, t_v denoting the computation time for forming the valid inequality (4.14) using the algorithm in Figure 4.1, forming the cut (4.64) using the algorithm in Figure 4.2 and solving the LP after adding (4.14) and (4.64). Average t_v for 88 instances is 629 seconds.

10.4. Location, Scheduling and Routing Problem

10.4.1. The Strength of the New Valid Inequality and Cutting Plane

In this section we test the strength of the valid inequalities proposed in Section 5.3. We generate 18 problem instances by varying the number of the grid points $J = m_1 \times m_2$ and the budget level B . We assign six different values to J ranging from 10 to 40. For each test instance we first solve the LP relaxation of LSRP (5.1) – (5.14) and let x_{LP}^1 and z_{LP}^1 denote respectively its optimal solution and objective value. We then add valid inequalities (5.15) – (5.18), (5.30), (5.32) to the original LP relaxation and solve the restricted LP relaxation. We let x_{LP}^2 and z_{LP}^2 denote, respectively, its optimal solution and objective value. We solve the LP relaxations using CPLEX 11.0 [62] with default options. Computational results are reported in Table 10.8. The first three columns contain the length of the rectangular grid, m_1 , the width of the rectangular grid, m_2 , and the budget level B . The third and fourth columns include the optimal values of the LP relaxations. As expected $z_{LP}^1 \geq z_{LP}^2$. Percent improvements (PI) introduced by the new valid inequalities are reported in the sixth column. These values are calculated

Table 10.6. Execution times for the first set of test problems

Instances	m	n	Dense		Sparse	
			t	t_v	t	t_v
SCP41	200	1000	1	243	0	67
SCP42	200	1000	1	243	0	29
SCP43	200	1000	1	241	0	28
SCP44	200	1000	1	231	0	29
SCP45	200	1000	1	482	0	29
SCP46	200	1000	1	243	0	29
SCP47	200	1000	1	243	0	29
SCP48	200	1000	1	240	0	29
SCP49	200	1000	1	240	0	29
SCP410	200	1000	1	243	0	29
SCP51	200	2000	1	1012	0	110
SCP52	200	2000	1	1028	0	181
SCP53	200	2000	1	921	0	106
SCP54	200	2000	1	974	0	103
SCP55	200	2000	1	917	0	104
SCP56	200	2000	1	918	0	106
SCP57	200	2000	1	917	0	105
SCP58	200	2000	1	973	0	199
SCP59	200	2000	1	973	0	118
SCP510	200	2000	1	974	0	121
SCP61	200	1000	1	254	0	77
SCP62	200	1000	1	239	0	44
SCP63	200	1000	1	239	0	43
SCP64	200	1000	1	225	0	40
SCP65	200	1000	1	237	0	41

Table 10.7. Execution times for the second set of test problems

Instances	m	n	Dense		Sparse	
			t	t_v	t	t_v
SCPA1	300	3000	2	3053	0	362
SCPA2	300	3000	2	3272	0	351
SCPA3	300	3000	2	3271	0	360
SCPA4	300	3000	2	3062	0	362
SCPA5	300	3000	2	3055	0	343
SCPB1	300	3000	2	3266	0	494
SCPB2	300	3000	2	3063	0	448
SCPB3	300	3000	2	3267	0	462
SCPB4	300	3000	2	3054	0	474
SCPB5	300	3000	2	3061	0	472
SCPE1	50	500	0	16	0	6
SCPE2	50	500	0	16	0	7
SCPE3	50	500	0	16	0	7
SCPE4	50	500	0	16	0	7
SCPE5	50	500	0	16	0	7
SCPC1R10	511	210	1	25	0	7
SCPC1R11	1023	330	2	117	0	38
SCPC1R12	2047	495	6	527	2	265
SCPC1R13	4095	715	26	2223	11	1251

according to

$$PI = 100 \times \frac{z_{LP}^1 - z_{LP}^2}{z_{LP}^1}. \quad (10.3)$$

Notice that positive values imply that x_{LP}^1 is cut off by one of the valid inequalities (5.15) – (5.18), (5.30), (5.32). The last two columns include respectively the solution times of the LP relaxations of the original (CPU¹) and the strengthened (CPU²) formulations. In all of the test instances valid inequalities (5.15) – (5.18), (5.30), (5.32)

Table 10.8. The strength of the new valid inequalities

m_1	m_2	B	Network Lifetime			Solution Time (sec)	
			z_{LP}^1	z_{LP}^2	PI (%)	CPU ¹	CPU ²
5	2	8	524	80	84.7	96	117
5	2	14	524	133	74.6	58	65
5	2	20	524	177	66.2	53	73
5	3	12	552	90	83.7	1418	1007
5	3	21	552	155	71.9	1300	958
5	3	30	552	214	61.2	1602	1805
5	4	15	581	90	84.5	7087	5038
5	4	28	581	165	71.6	6506	4799
5	4	40	581	225	61.2	8001	9018
5	5	19	607	94	84.5	21256	15105
5	5	35	607	172	71.7	19526	14405
5	5	50	607	243	60.0	23994	27061
6	5	23	632	98	84.5	42503	30206
6	5	42	632	179	71.7	39043	28814
6	5	60	632	250	60.4	47997	54115
10	4	30	656	102	84.5	63747	45315
10	4	55	656	179	72.7	58564	43214
10	4	80	656	260	60.4	72005	81166

cut off x_{LP}^1 (i.e. $PI > 0$ for all of them). They improve the upper bound provided

by the LP relaxation 84.4% on the average for instances with a low budget level. For instances with a medium budget level the improvement is 72.4% on the average. For instances with a high budget level the improvement is 61.6% on the average. These results point out that valid inequalities (5.15) – (5.18), (5.30), (5.32) increase remarkably the strength of the formulation.

10.4.2. Quality of the Upper Bound

In Chapter 6 we proposed a branch and price algorithm for LSRP (the algorithm in Figure 6.7). The branch and price algorithm solved optimally a sensor field that consists of 4 points (2×2 grid) for a planning horizon of 4 periods in 2534 seconds after evaluating 3390 nodes. However when the planning horizon is raised to 10 periods, the branch and price algorithm evaluated 8858 nodes in 2.5 days when we stopped program execution. It needed to evaluate more nodes. As this example shows since too many nodes appear the branch and price algorithm turns out to be prohibitively inefficient in finding the optimal solution for even small LSRP instances. However, the solution of the root node by the algorithm in Figure 6.1 provides a very good upper bound on the optimal value of LSRP as we show in this section.

We compare the upper bound provided by the algorithm in Figure 6.1 with that given by CPLEX 11.0 with the default options. The results are shown in Table 10.9. L_{UB}^* denotes the upper bound on the optimal objective provided by the algorithm in Figure 6.1. L_{UB}^C is the upper bound provided by CPLEX 11.0. In Table 10.9 A6.1 abbreviates the algorithm in Figure 6.1. In each problem instance we give CPLEX 11.0 much more time than the CPU time consumed by the algorithm in Figure 6.1. We see that the upper bound found by the algorithm in Figure 6.1 is better than that given by CPLEX 11.0 in 3 instances. They find the same upper bound in 6 instances. CPLEX 11.0 cannot provide an upper bound in 9 instances. The algorithm in Figure 6.1 is more efficient than CPLEX 11.0 in giving an upper bound. Average CPU time spent by the algorithm in Figure 6.1 is 4,216.7 seconds. Average CPU time spent by CPLEX 11.0 is 10,200 seconds.

Table 10.9. Comparison of the upper bounds for LSRP

m_1	m_2	B	Network Lifetime		CPU Time (s)	
			L_{UB}^*	L_{UB}^C	A6.1	CPLEX 11.0
5	2	8	77	77	235	3,600
5	2	14	130.5	130.5	224	3,600
5	2	20	174	174	128	3,600
5	3	12	87	87	1,263	7,200
5	3	21	152	152	1,586	7,200
5	3	30	211	-	4,260	7,200
5	4	15	87	91.5	1,176	10,800
5	4	28	162.5	164	1,794	10,800
5	4	40	223	-	7,624	10,800
5	5	19	91	95	1,853	10,800
5	5	35	169	-	2,347	10,800
5	5	50	240	-	5,663	10,800
6	5	23	95	-	3,726	14,400
6	5	42	175	-	7,450	14,400
6	5	60	247	-	12,031	14,400
10	4	30	98	98	5,220	14,400
10	4	55	174	-	7,140	14,400
10	4	80	256	-	12,180	14,400

In order to see the performance of the algorithm in Figure 6.1 in large LSRP instances we conduct further experiments. We vary $J = m_1 \times m_2$ from 60 to 400. The results are shown in Table 10.10. In these instances, although we give CPLEX 11.0 more time than the CPU time consumed by the algorithm in Figure 6.1, CPLEX 11.0 cannot provide an upper bound. We see that the algorithm in Figure 6.1 can provide an upper bound in all of the test instances. It consumes 38,798.4 seconds of CPU time on the average.

Table 10.10. Upper bounds for large LSRP instances

m_1	m_2	B	Network Lifetime	CPU Time (s)
			L_{UB}^*	
10	6	45	104	7,832
10	6	83	188	10,690
10	6	120	264	18,170
10	8	60	108	10,310
10	8	110	192	13,990
10	8	160	276	24,350
10	10	75	112	12,980
10	10	138	204	17,770
10	10	200	284	30,312
20	10	150	124	26,050
20	10	275	228	35,680
20	10	400	316	60,800
20	15	225	130	39,148
20	15	413	232	53,548
20	15	600	320	91,312
20	20	300	134	52,350
20	20	550	236	71,380
20	20	800	324	121,700

10.4.3. Accuracy and Efficiency of the New Heuristics

In this section we determine the accuracy and efficiency of the methods proposed to solve LSRP. In the first set of experiments we vary $J = m_1 \times m_2$ from 10 to 40. In some of these instances CPLEX 11.0 (run with the default options) provides a feasible solution. Table 10.11 shows the results. In this table $L^{A6.1}$ denotes the lifetime value corresponding to the feasible solution found by the algorithm in Figure 6.1. L^{LH} is the lifetime value found by the Lagrangean Heuristic (LH) presented in Chapter 7. L^{CGH} is the lifetime value found by the column generation based heuristic (CGH) presented

in chapter 8. L^{DSH} is the lifetime value found by the Disjoint Sets Heuristic (DSH) presented in Chapter 9. L^C is the lifetime value found by CPLEX 11.0 (C). UB is the upper bound on the optimal objective found in the previous section. In some of the instances CPLEX 11.0 cannot provide a feasible solution within the allowed time limit which is much greater than the CPU times consumed by the proposed methods. These instances are indicated by “-” in the table.

Table 10.11. Lifetime values found by the methods proposed for LSRP

m_1	m_2	B	Network Lifetime					
			$L^{A6.1}$	L^{LH}	L^{CGH}	L^{DSH}	L^C	UB
5	2	8	60	62	54	52	58	77
5	2	14	120	115	112	108	115.5	130.5
5	2	20	173.5	165	153	145	171.5	174
5	3	12	65	61	58	54	82	87
5	3	21	125	118	116	110	61.5	152
5	3	30	180	175	173	168	-	211
5	4	15	66	60	58	53	58	87
5	4	28	125	122	118	113	-	162.5
5	4	40	180	174	169	163	-	223
5	5	19	65	66	57	53	-	91
5	5	35	125	121	116	112	-	169
5	5	50	185	179	172	165	-	240
6	5	23	65	61	57	52	-	95
6	5	42	125	120	114	108	-	175
6	5	60	235	240	229	220	-	247
10	4	30	66	61	56	50	-	98
10	4	55	125	116	112	103	-	174
10	4	80	235	228	224	211	-	256

Table 10.12 shows the accuracies of the methods proposed to solve LSRP. As in the section where we determine the accuracy of the LSP we define the accuracy as

the percent deviation from the upper bound, which is calculated using formula (10.1). We see that the most accurate method is the algorithm in Figure 6.1. DSH is the least accurate method. However, as we will see in the next table it consumes the least amount of CPU time.

Table 10.12. Accuracies of the methods proposed to solve LSRP

m_1	m_2	B	Accuracy				
			A6.1	LH	CGH	DSH	C
5	2	8	22.1	19.5	29.9	32.5	24.7
5	2	14	8.0	11.9	14.2	17.2	11.5
5	2	20	0.3	5.2	12.1	16.7	1.4
5	3	12	25.3	29.9	33.3	37.9	5.7
5	3	21	17.8	22.4	23.7	27.6	59.5
5	3	30	14.7	17.1	18	20.4	-
5	4	15	24.1	31.0	33.3	39.1	33.3
5	4	28	23.1	24.9	27.4	30.5	-
5	4	40	19.3	22.0	24.2	26.9	-
5	5	19	28.6	27.5	37.4	41.8	-
5	5	35	26.0	28.4	31.4	33.7	-
5	5	50	22.9	25.4	28.3	31.3	-
6	5	23	31.6	35.8	40	45.3	-
6	5	42	28.6	31.4	34.9	38.3	-
6	5	60	4.9	2.8	7.3	10.9	-
10	4	30	32.7	37.8	42.9	49.0	-
10	4	55	28.2	33.3	35.6	40.8	-
10	4	80	8.2	10.9	12.5	17.6	-
Average			20.3	23.2	27.0	31.0	22.7

Table 10.13 shows the CPU times consumed by the methods proposed to solve LSRP. We realize that accuracy has a price. The most accurate method which is the algorithm in Figure 6.1 consumes the largest CPU time on the average. DSH being

the least accurate method consumes much smaller CPU time than the other methods.

Table 10.13. CPU Time consumed by the methods proposed to solve LSRP

m_1	m_2	B	CPU Time (s)				
			A6.1	LH	CGH	DSH	C
5	2	8	235	492	8	0.2	3,600
5	2	14	224	1,189	11	0.4	3,600
5	2	20	128	5	10	0.6	3,600
5	3	12	1,263	1	90	0.2	7,200
5	3	21	1,586	683	106	0.4	7,200
5	3	30	4,260	3,600	46	0.6	7,200
5	4	15	1,176	3,600	276	0.2	10,800
5	4	28	1,794	3,600	294	0.4	10,800
5	4	40	7,624	3,601	216	0.6	10,800
5	5	19	1,853	3,600	1040	0.2	10,800
5	5	35	2,347	3,600	846	0.4	10,800
5	5	50	5,663	379	288	0.5	10,800
6	5	23	3,726	3,601	4836	0.4	14,400
6	5	42	7,450	5,003	618	0.6	14,400
6	5	60	12,031	252	245	0.8	14,400
10	4	30	5,220	3,601	15,268	0.5	14,400
10	4	55	7,140	6,001	3,274	0.7	14,400
10	4	80	12,180	294	3,224	1.0	14,400
Average			4216.7	2394.6	1705.3	0.5	10,200

We conduct further experiments in order to see the performance of the proposed methods in large LSRP instances. We vary $J = m_1 \times m_2$ from 60 to 400. Table 10.14 shows the lifetime values found. In these instances, although we give CPLEX 11.0 much more time than the CPU time consumed by the proposed methods, CPLEX 11.0 does not provide a feasible solution.

Table 10.14. Lifetime values found by the methods proposed for LSRP on large instances

m_1	m_2	B	Network Lifetime				
			$L^{A6.1}$	L^{LH}	L^{CGH}	L^{DSH}	UB
10	6	45	65	60	56	48	104
10	6	83	175	171	168	153	188
10	6	120	236	228	224	212	264
10	8	60	69	59	56	49	108
10	8	110	175	179	168	159	192
10	8	160	236	230	224	212	276
10	10	75	68	60	56	49	112
10	10	138	177	180	168	159	204
10	10	200	236	223	224	212	284
20	10	150	65	60	56	51	124
20	10	275	175	172	168	158	228
20	10	400	236	225	224	212	316
20	15	225	65	59	56	48	130
20	15	413	180	175	168	157	232
20	15	600	238	230	224	210	320
20	20	300	66	60	56	50	134
20	20	550	176	171	168	161	236
20	20	800	238	237	224	210	324

Accuracies of the methods on large instances are presented in Table 10.15. We see again that the most accurate method is the algorithm in Figure 6.1. The least accurate method is DSH. However, as the next table shows DSH is much faster than the other methods.

Table 10.16 shows the CPU times consumed by the proposed methods to solve LSRP on large instances. DSH is much faster than the other methods. Algorithm in Figure 6.1 being the most accurate method is faster than LH and DSH on the average.

Table 10.15. Accuracies of the methods proposed to solve LSRP on large instances

m_1	m_2	B	Accuracy			
			A6.1	LH	CGH	DSH
10	6	45	37.5	42.3	46.2	53.8
10	6	83	6.9	9.0	10.6	18.6
10	6	120	10.6	13.6	15.2	19.7
10	8	60	36.1	45.4	48.1	54.6
10	8	110	8.9	6.8	12.5	17.2
10	8	160	14.5	16.7	18.8	23.2
10	10	75	39.3	46.4	50.0	56.3
10	10	138	13.2	11.8	17.6	22.1
10	10	200	16.9	21.5	21.1	25.4
20	10	150	47.6	51.6	54.8	58.9
20	10	275	23.2	24.6	26.3	30.7
20	10	400	25.3	28.8	29.1	32.9
20	15	225	50.0	54.6	56.9	63.1
20	15	413	22.4	24.6	27.6	32.3
20	15	600	25.6	28.1	30.0	34.4
20	20	300	50.7	55.2	58.2	62.7
20	20	550	25.4	27.5	28.8	31.8
20	20	800	26.5	26.9	30.9	35.2
Average			26.7	29.7	32.4	37.4

10.4.4. Experiments With Very Large Instances

Based on the previous subsection we can say that DSH is the most efficient and compute a feasible solution of LSRP for very large WSNs in a meaningful time. Hence, we conduct additional experiments using DSH on very large instances, i.e. for sensor fields with upto 2000 points.

Table 10.16. CPU Time consumed by the methods proposed to solve LSRP in large-sized instances

m_1	m_2	B	CPU Time (s)			
			A6.1	LH	CGH	DSH
10	6	45	7,832	6,001	22,250	0.7
10	6	83	10,690	590	45,676	1
10	6	120	18,170	898	9,041	1.5
10	8	60	10,310	9,009	55,906	0.9
10	8	110	13,990	3,185	48,329	1.3
10	8	160	24,350	2,543	11,282	2
10	10	75	12,980	17,002	68,901	1.2
10	10	138	17,770	6,037	33,239	1.7
10	10	200	30,312	7,065	30,503	2.5
20	10	150	26,050	736	139,480	2.3
20	10	275	35,680	41,001	56,096	3.3
20	10	400	60,800	58,037	41,701	5
20	15	225	39,148	59,007	209,220	3.5
20	15	413	53,548	101,002	84,143	5
20	15	600	91,312	127,003	62,551	7.5
20	20	300	52,350	62,473	278,960	4.7
20	20	550	71,380	175,096	112,191	6.7
20	20	800	121,700	216,010	83,401	10
Average			38,798.4	49,594.2	77,381.7	3.4

10.4.4.1. The Performance of the Disjoint Sets Heuristic. Additional experiments are conducted to see the performance of DSH on very large WSN instances where the number of grid points varies from 200 to 2000. Tables 10.17 and 10.18 contain information about the composition and active time of the disjoint sets found by DSH as well as the network lifetime L^{DSH} . In particular, we present the number of type-1 and type-2 sensors deployed, and the active time $L_{\Psi_n}^*$ corresponding to each disjoint sensor set Ψ_n . Note that L^{DSH} is the sum of $L_{\Psi_n}^*$'s of all sensor sets. As can be observed in Table

10.17 and Table 10.18, for a given network size at least one new connected disjoint set, which covers the grid points (i.e. sensor field) with the required quality and costs less than the additional budget, is found by our heuristic as the budget increases from the low to the medium level or from the medium to the high level.

Figure 10.1 illustrates the lifetime values (in days) obtained by DSH for each test instance at different budget levels which are given in the last column of Table 10.17 and Table 10.18. We observe that for a given network size the lifetime increases in all ten problems as the budget level increases from low to medium or from medium to high. Figure 10.2 shows the CPU time required by DSH for each test instance reported in Table 10.17 and Table 10.18. Using these CPU times after dividing by N , we also perform a polynomial regression analysis to find out the relationship between the average CPU time to construct a disjoint set Ψ_n and the network size in terms of the number of points. The results are presented in Table 10.19. The first column shows the order of the polynomial (OP) used in the regression. The third column contains the equation of the fitted polynomial. The value of the R -squared statistic, a measure of the adequacy of the fitted polynomial to explain the relationship, is given in the last column. We notice that a second order polynomial is adequate to predict the CPU time of building a disjoint set for a given network size. Improvements in R -squared values are very small when the order of the polynomial is raised to three. Since at most JK disjoint sets can be formed by DSH, this regression analysis implies a third order variation in the CPU time of DSH as a function of the network size. Formula (9.13) is based on the worst-case complexity analysis. For example, the third term in (9.13), which is the complexity of solving an LPFG instance, is much higher than the average complexity of a simplex-based solver such as CPLEX 11.0. Therefore, this analysis also validates the worst-case complexity formula that shows the fourth order variation in the CPU time as a function of the number of points, J .

10.4.4.2. The Effect of Data Packet Generation Rate on the Lifetime. We conduct further experiments to analyze the effect of data packet generation rate (h) of the sensors on the lifetime. The network size is fixed at $20 \times 15 = 300$ points. Figure 10.3 contains

Table 10.17. Composition and active times of disjoint sets for the first set of experiments

m_1	m_2	B	# Type-1, # Type-2, Active Time of Each Ψ_n				L^{DSH}
			Ψ_1	Ψ_2	Ψ_3	Ψ_4	
20	10	150	20,30,57	-	-	-	51
20	10	275	20,30,57	19,31,57	17,33,57	-	158
20	10	400	20,30,57	19,31,57	17,33,57	19,34,57	212
20	20	300	73,51,57	-	-	-	50
20	20	550	73,51,57	65,57,57	44,68,57	-	161
20	20	800	73,51,57	65,57,57	44,68,57	40,71,57	210
30	20	450	108,76,57	-	-	-	57
30	20	825	108,76,57	92,88,56	73,97,56	-	169
30	20	1200	108,76,57	92,88,56	73,97,56	60,104,57	226
40	20	600	138,100,56	-	-	-	56
40	20	1100	138,100,56	121,115,56	93,132,57	-	169
40	20	1600	138,100,56	121,115,56	93,132,57	91,131,55	224
50	20	750	169,126,55	-	-	-	55
50	20	1375	169,126,55	154,142,56	110,159,56	-	167
50	20	2000	169,126,55	154,142,56	110,159,56	108,165,56	223
50	24	900	196,152,53	-	-	-	53
50	24	1650	196,152,53	186,164,56	150,182,56	-	165
50	24	2400	196,152,53	186,164,56	150,182,56	125,195,56	221
50	28	1050	223,180,51	-	-	-	51
50	28	1925	223,180,51	200,198,55	163,211,56	-	162
50	28	2800	223,180,51	200,198,55	163,211,56	139,234,56	218
50	32	1200	276,204,54	-	-	-	54
50	32	2200	276,204,54	222,226,54	203,241,55	-	163
50	32	3200	276,204,54	222,226,54	203,241,55	178,256,55	218

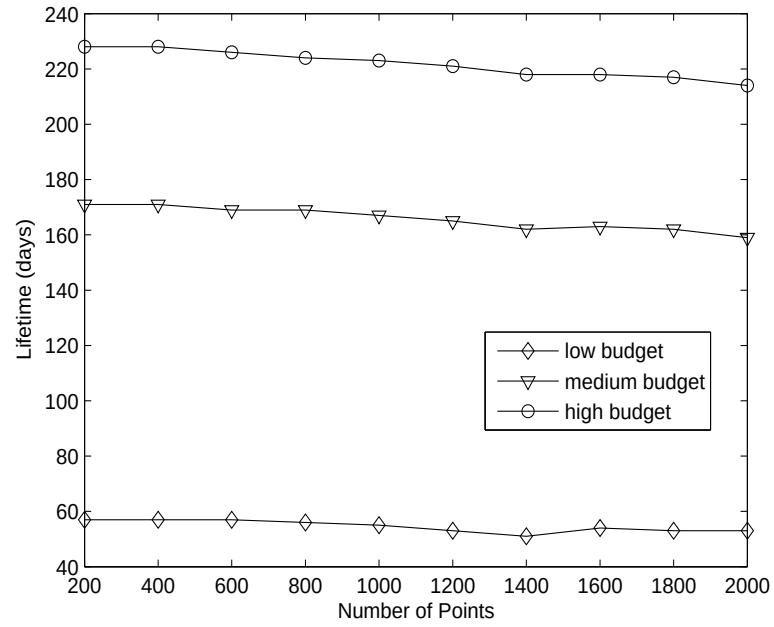


Figure 10.1. Lifetime as a function of the number of points.

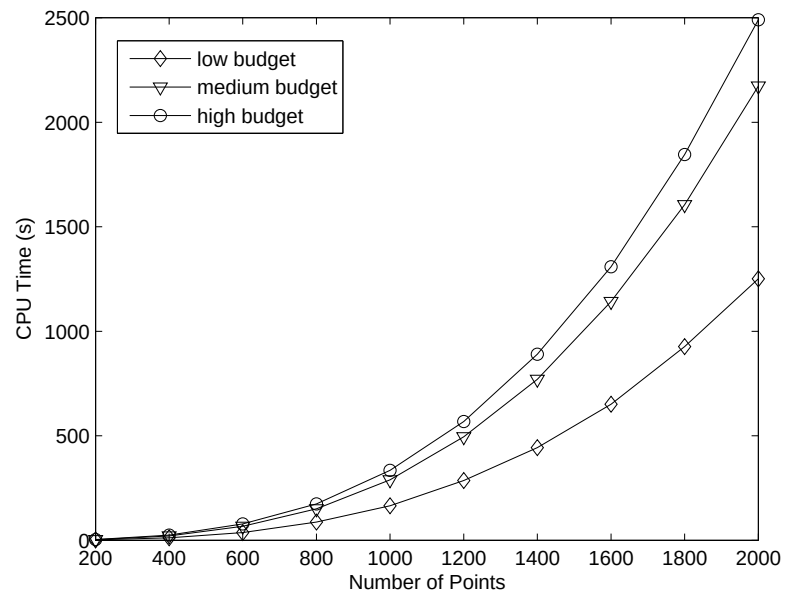


Figure 10.2. CPU time as a function of the number of points.

Table 10.18. Composition and active times of disjoint sets for the second set of experiments

m_1	m_2	B	# Type-1, # Type-2, Active Time of Each Ψ_n				L^{DSH}
			Ψ_1	Ψ_2	Ψ_3	Ψ_4	
50	36	1350	312,230,53	-	-	-	53
50	36	2475	312,230,53	241,257,53	222,273,56	-	162
50	36	3600	312,230,53	241,257,53	222,273,56	174,292,55	217
50	40	1500	331,255,53	-	-	-	53
50	40	2750	331,255,53	275,277,51	222,310,55	-	159
50	40	4000	331,255,53	275,277,51	222,310,55	233,311,55	214

Table 10.19. Regression analysis for the relation between the CPU time of DSH and the number of points.

OP	Budget	CPU Time as a Function of J	R^2
1	low	$0.4180J$	71.78
1	medium	$0.2423J$	71.91
1	high	$0.2087J$	72.04
2	low	$0.00003492J^2 - 0.2565J$	99.35
2	medium	$0.00002328J^2 - 0.1473J$	99.37
2	high	$0.00002168J^2 - 0.1256J$	99.39
3	low	$0.0000001664J^3 + 0.00003492J^2 - 0.009761J$	100.00
3	medium	$0.00000007824J^3 + 0.00002328J^2 - 0.006736J$	100.00
3	high	$0.00000009401J^3 + 0.00002168J^2 - 0.006020J$	100.00

the plots of the lifetime values found as the data packet generation rate increases from 24 to 1440 packets per period. This corresponds to a range of data generation frequencies starting with the case in which a data packet is generated every half an hour ($2 \text{ packets/hour} \times 12 \text{ hours} = 24 \text{ packets/period}$) up to the case in which a packet is generated every 30 seconds ($120 \text{ packets/hour} \times 12 \text{ hours} = 1440 \text{ packets/period}$). We notice that the network lifetime decreases significantly with an increase in the data packet generation rate at all budget levels. This is due to the increased energy consumption of the sensors in routing the data packets to the sinks. That is, increasing the

packet generation rate causes a larger amount of data to be transmitted to the sinks. Therefore, we can conclude that the design of a WSN without considering the routing issue, i.e. a design attempt without integration, may result in erroneous estimations for the network lifetime.

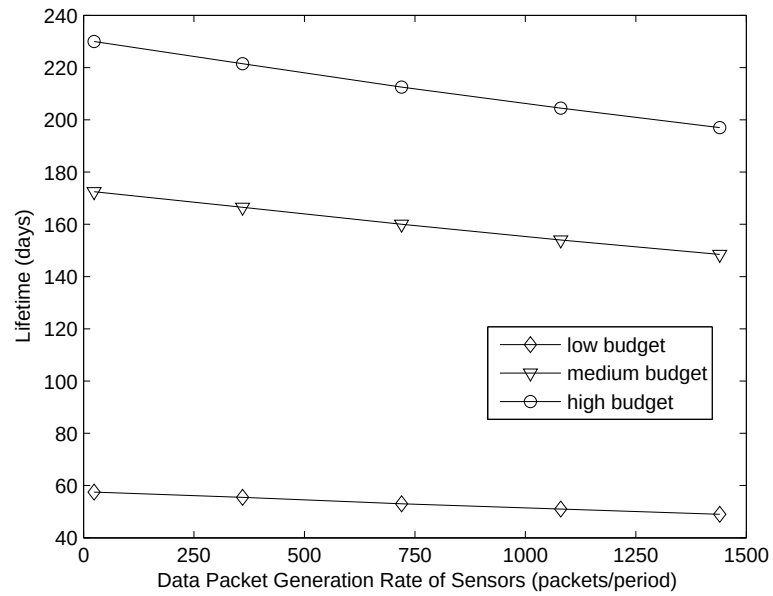


Figure 10.3. Lifetime as a function of the data packet generation rate of sensors.

11. CONCLUSIONS

In this thesis we first worked on the Location and Scheduling Problem. After formulating a MILP for the problem we developed a Lagrangean Heuristic. Experiments show that the proposed Lagrangean Heuristic is more accurate than the commercial solver and provides a feasible solution and an upper bound for large instances, for which a commercial solver can provide neither a feasible solution nor an upper bound.

We also proposed a new valid inequality and a new cutting plane for the covering integer programs. For the valid inequality we defined a sufficient condition for facet definition and proposed a new procedure that gives the valid inequality using lifting. Experiments show that the new valid inequality and the cutting plane cut off LP relaxation solution and improve considerably the lower bound obtained.

We then consider the integrated location, scheduling and routing problem and proposed a MILP formulation, which we strengthened by developing new valid inequalities. Experiments show that the new valid inequalities improve considerably the upper bound provided by the LP relaxation solution. Next, we proposed a branch and price algorithm that computes an optimal solution and three heuristic procedures. It is based on the reformulation of the problem. Experiments show that the branch and price algorithm is inefficient. However, the solution of the root node gives a very good upper bound on the optimal objective of LSRP. The upper bound obtained by solving the root node is stronger than the one given by a commercial solver in all of the test instances. When we compare the feasible solutions found we see that the reformulation based heuristic is the most accurate one. Lagrangean and column generation heuristics are respectively in the second and third ranks with respect to their accuracy. DSH is the least accurate method, but it is much more efficient and it makes possible the computation of a feasible solution even for extremely large instances.

In this thesis experiments are conducted in order to determine the effect of data packet generation rate on the network lifetime. We notice that the network lifetime

decreases significantly with an increase in the data packet generation rate at all budget levels. We can say that the design of a WSN without considering the routing issue, i.e. a design attempt without integration, may result in erroneous estimations for the network lifetime.

As a future research direction one can develop an efficient solution procedure for the exact solution of LSP. This can be based on column generation or branch and price. One can also put the column generation method proposed in the thesis in a metaheuristic and can find better results for LSRP with a more succesful search. In additon it is a good idea to add connectivity constraints to the LSRP formulation. However this would make the solution method harder because the number of constraints grow exponentially with the problem size. One can also add bandwidth allocation constraints to the LSRP formulation. However this would make the problem nonlinear.

APPENDIX A: ON THE RELATION BETWEEN COVERAGE AND CONNECTIVITY

Zhang and Hou [63] show that in an homogenous sensor network if the transmission range is greater than or equal to two times the sensing range coverage of the sensor field implies connectivity of the sensors. In this appendix we show by a counter example that this theorem does not necessarily hold in a heterogenous sensor network.

In the counter example coverage quality requirement is two for each point of the sensor field, i.e. each point of the sensor field should be within the sensing range of two different sensors. In the counter example sensor specifications are the same as in Table 10.1. As we see in this table transmission range of each sensor type is two times the sensing range. When these sensors are placed in a 20×20 grid as shown in Figure A.1 coverage of the sensor field which is the 20×20 grid is satisfied. As we see, each point of the grid is within the sensing range of two different sensors. However, all of the sensors are not connected. The two adjacent type-1 sensors that are placed at the bottom right of the sensor field cannot transmit the data that they collect to any other sensors because their transmission range is not sufficient.

The counter example in Figure A.1 proves that the theorem proposed by Zhang and Hou is not valid in an heterogenous sensor network. Coverage of the sensor field does not necessarily imply connectivity of the sensors even if transmission range is greater than or equal to two times the sensing range.

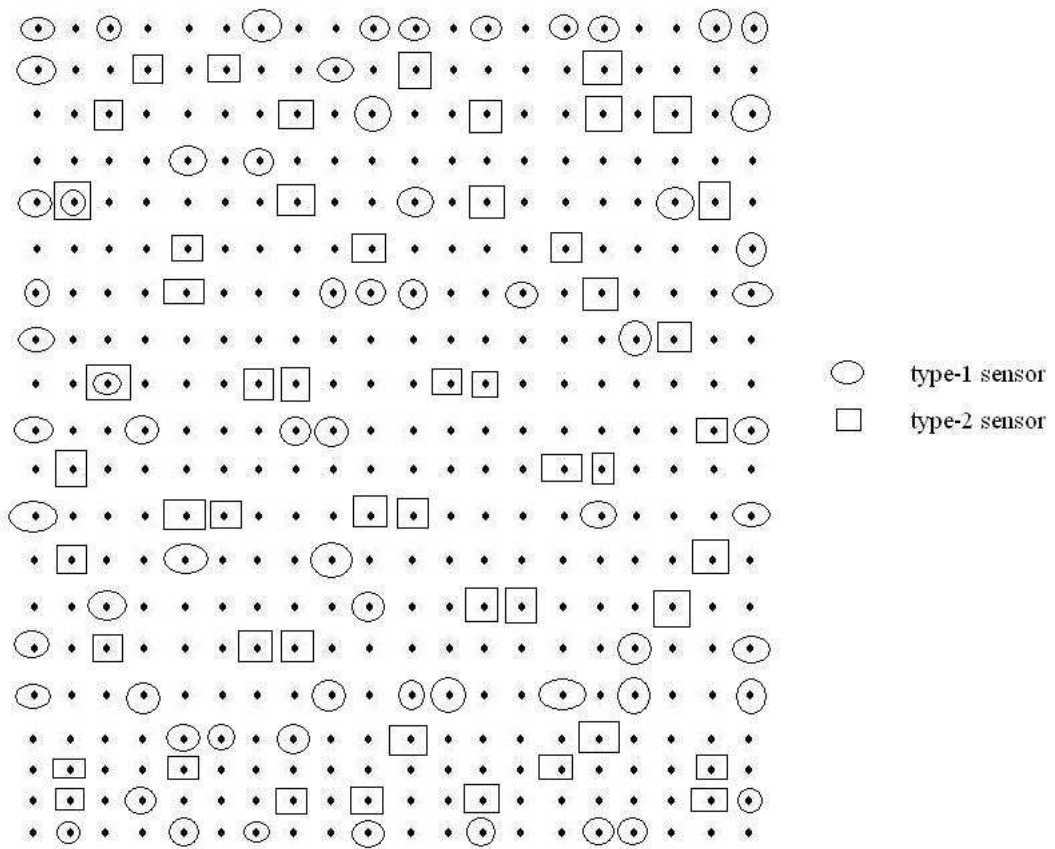


Figure A.1. A disconnected sensor placement that covers the sensor field

APPENDIX B: ON THE CONVERGENCE OF THE DEFLECTED SUBGRADIENT ALGORITHM

Consider the following optimization problem

$$\text{supremum } \{f(x) : Ax = b, Cx \leq d, x \in h \subset \mathbb{R}^n\} . \quad (\text{B.1})$$

In (B.1) $f(x)$ is a linear function from $\mathbb{R}^n$ to $\mathbb{R}$. h is a nonempty, linear, bounded, mixed integer set in $\mathbb{R}^n$. A linear mixed-integer set is the one defined by intersections of half-spaces and hyperplanes in $\mathbb{R}^n$ and some components of the set's elements are restricted to be integer. In (B.1) A is a real matrix having r_A rows and n columns. b is an r_A dimensional real vector. In (B.1) C is a real matrix having r_C rows and n columns. d is an r_C dimensional real vector. We assume that $r_A \geq 1$ or $r_C \geq 1$ or both $r_A, r_C \geq 1$. For an $u \in D$ where

$$D = \left\{ u \in \mathbb{R}^{r_A+r_C} \text{ such that } u_i \in \mathbb{R} \ i \in \{1, 2, \dots, r_A\} \text{ and } u_i \in \mathbb{R}^+ \right. \\ \left. i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\} \right\} , \quad (\text{B.2})$$

the following problem is called the Lagrangean Problem.

$$\phi(u) = \sup \left\{ f(x) + \sum_{i=1}^{r_A} u_i (b_i - (Ax)_i) + \sum_{i=r_A+1}^{r_A+r_C} u_i (d_i - (Cx)_i) : x \in h \right\} \quad (\text{B.3})$$

Claim 1. *In (B.3) supremum (the greatest lower bound) exists and occurs at least at a point $x \in h$.*

Proof. Since h is a nonempty, linear, bounded, mixed-integer set in $\mathbb{R}^n$, it can be showed that its convex hull is a non-empty polytope P in $\mathbb{R}^n$. It can also be showed

that a linear function defined from $\mathbb{R}^n$ to $\mathbb{R}$ such as

$$f(x) + \sum_{i=1}^{r_A} u_i (b_i - (Ax)_i) + \sum_{i=r_A+1}^{r_A+r_C} u_i (d_i - (Cx)_i) \quad (\text{B.4})$$

has a maximum at one extreme point of P . And extreme points of P are elements of h . See [58] for the proofs of these arguments. Q.E.D.

In this thesis chapter a rigorous proof for the convergence of the deflected subgradient algorithm that switches to the conditional subgradient algorithm when the deflected direction is infeasible at the boundary is presented. The purpose is to find the flaws of the recent work by Guta [66]. For instance, a subsequence of a sequence can converge to a point even if the sequence does not. This fact is omitted in the convergence proofs. Also, two or three dimensional figures are not sufficient to prove rules in $\mathbb{R}^n$ for $n \geq 4$.

Claim 2. D is a non-empty, closed, convex set in $\mathbb{R}^{r_A+r_C}$.

Proof. $0 \in D$ so D is a non-empty set. Given $u^1, u^2 \in D$, $\alpha u^1 + (1 - \alpha) u^2 \in D$ for $\alpha \in [0, 1]$ since convex combinations of non-negative components are non-negative. Therefore D is convex. Let's assume that D is not closed. This implies that there is a sequence $x^k, k \geq 0$ in D that converges to a point x^* outside of D . x^* should have at least a component $x_{i^*}^*, i^* \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}$ which is less than 0 since it is outside of D . Consider the closed ball $B(x^*, 0.5 * |x_{i^*}^*|)$ in $\mathbb{R}^{r_A+r_C}$. A point x^k cannot be an element of this ball because $x_{i^*}^k \geq 0$. This contradicts convergence of x^k to x^* . Q.E.D.

Claim 3. Given a point y in $\mathbb{R}^{r_A+r_C}$, projection of y on D denoted as $P_D(y)$ is

$$P_D(y) = \begin{cases} y_i & i \in \{1, 2, \dots, r_A\} \\ \max(0, y_i) & i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\} \end{cases} \quad (\text{B.5})$$

Proof. Consider the following nonlinear optimization problem:

$$\text{infemum } \{ \|x - y\|^2 : x \in D \}. \quad (\text{B.6})$$

We can rewrite D as

$$D = \{ z \in \mathbb{R}^{r_A+r_C} : z_i \geq 0 \ i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\} \}. \quad (\text{B.7})$$

Because D is a non-empty, closed, convex set as proved in claim 2, by a fundamental property of the projection operator, infemum in (B.6) is a minimum. Due to another fundamental property of the projection operator, $x^* \in D$ giving that minimum is unique [79]. x^* can also be defined as the optimum solution of the following non-linear program

$$\begin{aligned} \min \quad & \sum_{i=1}^{r_A+r_C} (x_i - y_i)^2 \\ \text{s.t.} \quad & -x_i \leq 0 \ i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\} \\ & x \in \mathbb{R}^{r_A+r_C}. \end{aligned} \quad (\text{B.8})$$

In (B.8) objective and constraint functions are continuously differentiable and gradients of the constraint functions are linearly independant at $\forall x \in \mathbb{R}^{r_A+r_C}$. Therefore Karush-Kuhn-Tucker first order conditions [80] have to be satisfied at x^* , i.e. there exists a unique v vector in $\mathbb{R}^{r_C}$ with

$$2(x_i^* - y_i) = 0 \ i \in \{1, 2, \dots, r_A\} \quad (\text{B.9})$$

$$2(x_i^* - y_i) - v_i = 0 \ i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\} \quad (\text{B.10})$$

$$-x_i^* v_i = 0 \ i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\} \quad (\text{B.11})$$

$$-x_i^* \leq 0 \ i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\} \quad (\text{B.12})$$

$$v_i \geq 0 \ i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\} \quad (\text{B.13})$$

(B.9) implies

$$x_i^* = y_i \quad i \in \{1, 2, \dots, r_A\}. \quad (\text{B.14})$$

(B.10), (B.13) imply

$$x_i^* \geq y_i \quad i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}. \quad (\text{B.15})$$

For $i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}$ if $y_i > 0$ then from (B.15) $x_i^* > 0$. Therefore $v_i = 0$ from (B.11). (B.10) implies

$$x_i^* = y_i \quad \text{for } i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\} \quad \text{if } y_i > 0. \quad (\text{B.16})$$

For $i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}$ if $y_i \leq 0$, suppose $x_i^* > 0$. This implies with (B.10) that $v_i > 0$. But this means $x_i^* = 0$ from (B.11). This is a contradiction. Therefore,

$$x_i^* \leq 0 \quad \text{for } i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\} \quad \text{if } y_i \leq 0. \quad (\text{B.17})$$

(B.17) and (B.12) imply

$$x_i^* = 0 \quad \text{for } i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\} \quad \text{if } y_i \leq 0. \quad (\text{B.18})$$

In summary,

$$x_i^* = y_i \quad i \in \{1, 2, \dots, r_A\} \quad (\text{B.19})$$

$$x_i^* = y_i \quad \text{for } i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\} \quad \text{if } y_i > 0 \quad (\text{B.20})$$

$$x_i^* = 0 \quad \text{for } i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\} \quad \text{if } y_i \leq 0. \quad (\text{B.21})$$

A vector $s \in \mathbb{R}^{r_A + r_C}$ having the following property at a point $\bar{u} \in D$

$$\phi(u) \geq \phi(\bar{u}) + (u - \bar{u})^T s \quad \text{for } u \in D \quad (\text{B.22})$$

is called a subgradient of ϕ at $\bar{u}$.

Q.E.D.

Claim 4. Let $\bar{x}$ denote the solution of Lagrangean problem (B.3) giving $\phi(\bar{u})$ for a $\bar{u} \in D$. The vector s defined as

$$s_i = \begin{cases} b_i - (A\bar{x})_i & i \in \{1, 2, \dots, r_A\} \\ d_i - (C\bar{x})_i & i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\} \end{cases} \quad (\text{B.23})$$

is a subgradient of ϕ at $\bar{u}$.

Proof.

$$\phi(\bar{u}) = f(\bar{x}) + \sum_{i=1}^{r_A} \bar{u}_i (b_i - (A\bar{x})_i) + \sum_{i=r_A+1}^{r_A+r_C} \bar{u}_i (d_i - (C\bar{x})_i) \quad (\text{B.24})$$

since $\bar{x}$ is an optimal solution of Lagrangean problem (B.3) giving $\phi(\bar{u})$.

$$\phi(u) \geq f(\bar{x}) + \sum_{i=1}^{r_A} u_i (b_i - (A\bar{x})_i) + \sum_{i=r_A+1}^{r_A+r_C} u_i (d_i - (C\bar{x})_i) \quad \forall u \in D \quad (\text{B.25})$$

since $\bar{x}$ is only an element of h in the problem (B.3) giving $\phi(u)$. When we subtract each side of (B.24) from (B.25) we obtain

$$\phi(u) \geq \phi(\bar{u}) + \sum_{i=1}^{r_A} (u_i - \bar{u}_i) (b_i - (A\bar{x})_i) + \sum_{i=r_A+1}^{r_A+r_C} (u_i - \bar{u}_i) (d_i - (C\bar{x})_i) \quad \forall u \in D. \quad (\text{B.26})$$

(B.26) can equivalently be written as

$$\phi(u) \geq \phi(\bar{u}) + (u - \bar{u})^T \begin{bmatrix} b - A\bar{x} \\ d - C\bar{x} \end{bmatrix} \quad \forall u \in D. \quad (\text{B.27})$$

Therefore (B.23) is a subgradient of ϕ at $(\bar{u})$.

Q.E.D.

Claim 5. *The vector s defined in (B.23) has the following property:*

$$\exists \omega \in \mathbb{R}^+ \text{ such that } \|s\| \leq \omega \quad \forall \bar{x} \in h \quad (\text{B.28})$$

Proof. Since h is a bounded set, $\exists \gamma \in \mathbb{R}^+$ such that $|\bar{x}_i| \leq \gamma \quad \forall i \in \{1, 2, \dots, n\}$. Let ν represent the greatest real number in matrices A and C and in vectors b and d . Notice that there is a finite number of elements in those vectors and matrices by their definitions. Therefore ν exists. Using these arguments and the definition of s in (B.23) we can state that

$$|s_i| \leq n \times \gamma \times \nu + \nu \quad \forall i \in \{1, 2, \dots, r_A + r_C\}.$$

For any u element of D normal cone $N_D(u)$ is

$$N_D(u) = \{y \in \mathbb{R}^{r_A+r_C} : y^T(z - u) \leq 0 \quad \forall z \in D\} \quad (\text{B.29})$$

Q.E.D.

Claim 6. *Given s a subgradient at $\bar{u}$ and $y \in N_D(\bar{u})$. $(s + y)$ is also a subgradient vector at $\bar{u}$.*

Proof.

$$\phi(u) \geq \phi(\bar{u}) + (u - \bar{u})^T s \quad \forall u \in D \quad (\text{B.30})$$

since s is a subgradient at $\bar{u}$ (B.22).

$$\phi(u) \geq \phi(\bar{u}) + (u - \bar{u})^T s + y^T(u - \bar{u}) \quad \forall u \in D \quad (\text{B.31})$$

since from (B.29) $y^T(u - \bar{u}) \leq 0 \forall u \in D$. Factorising (B.31) we obtain

$$\phi(u) \geq \phi(\bar{u}) + (u - \bar{u})^T(s + y) \quad \forall u \in D. \quad (\text{B.32})$$

Q.E.D.

Claim 7. $N_D(u)$ in $\mathbb{R}^{(r_A+r_C)}$ (B.29) is a nonempty, closed, convex set.

Proof. $0 \in \mathbb{R}^{(r_A+r_C)}$ and $0^T(z - u) \leq 0 \forall z \in D$. Therefore $N_D(u)$ is non-empty. For any two $y^1, y^2 \in N_D(u)$

$$(y^1)^T(z - u) \leq 0 \quad \forall z \in D \quad (\text{B.33})$$

$$(y^2)^T(z - u) \leq 0 \quad \forall z \in D \quad (\text{B.34})$$

from the definition of $N_D(u)$ (B.29). Multiplying (B.33) by α and (B.34) by $(1 - \alpha)$ for $\alpha \in [0, 1]$ and summing the corresponding sides of both inequalities we obtain

$$\left(\alpha (y^1)^T + (1 - \alpha) (y^2)^T \right) (z - u) \leq 0 \quad \forall z \in D. \quad (\text{B.35})$$

Therefore $N_D(u)$ is a convex set. Suppose $N_D(u)$ is not closed. This implies that there exists a sequence $y^s, s \geq 0$ in $N_D(u)$ that converges to $\bar{y}$ not in $N_D(u)$. $\bar{y}$ is not in $N_D(u)$ implies

$$(\bar{y})^T(\bar{z} - u) = \varepsilon^1 > 0 \quad \text{for a } \bar{z} \in D. \quad (\text{B.36})$$

$$(y^s)^T(\bar{z} - u) \leq 0 \quad \forall s \geq 0 \text{ since } y^s \in N_D(u) \quad \forall s \geq 0 \quad (\text{B.37})$$

$$(B.36), (B.37) \text{ imply } (y^s - \bar{y})^T(\bar{z} - u) \leq -\varepsilon^1 < 0 \quad \forall s \geq 0 \quad (\text{B.38})$$

$$(B.38) \text{ implies } \left| (y^s - \bar{y})^T(\bar{z} - u) \right| \geq \varepsilon^1 > 0 \quad \forall s \geq 0 \quad (\text{B.39})$$

$$\text{Cauchy-Schwarz inequality implies } \left| (y^s - \bar{y})^T(\bar{z} - u) \right| \leq \|y^s - \bar{y}\| \|\bar{z} - u\| \quad \forall s \geq 0 \quad (\text{B.40})$$

$$(B.39), (B.40) \text{ imply } 0 < \varepsilon^1 \leq \|y^s - \bar{y}\| \|\bar{z} - u\| \quad \forall s \geq 0. \quad (\text{B.41})$$

If $\bar{z}$ is equal to u then $0 < \varepsilon^1 \leq 0$ from (B.41). This is a contradiction. If $\bar{z}$ is not equal to u , $y^s \rightarrow \bar{y}$ implies

$$\exists N \geq 0 \forall s \geq N \quad \|y^s - \bar{y}\| \leq \frac{\varepsilon^1}{2\|\bar{z} - u\|}. \quad (\text{B.42})$$

(B.41), (B.42) imply $0 < \varepsilon^1 \leq \frac{\varepsilon^1}{2}$. This is also a contradiction. Q.E.D.

Claim 8. *Given s a vector in $\mathbb{R}^{r_A+r_C}$, $u \in D$, $y^* = P_{N_D(u)}(s)$*

$$y_i^* = \begin{cases} s_i & \text{if } i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}, s_i < 0, u_i = 0 \\ 0 & \text{otherwise} \end{cases}$$

Proof. By definition of projection of a point on a set we are after the optimal solution y^* of the following mathematical program:

$$\text{Min} \quad \|s - y\|^2 \quad (\text{B.43})$$

s.t.

$$y^T(z - u) \leq 0 \quad \forall z \in D \quad (\text{B.44})$$

$$y \in \mathbb{R}^{r_A+r_C} \quad (\text{B.45})$$

Note that $N_D(u)$ is a non-empty, closed, convex set as proved in Claim 7. By projection property [79] y^* exists and it is unique. For $i \in \{1, 2, \dots, r_A\}$ assume $y_i^* \neq 0$, choose

$$z_j^* = \begin{cases} u_j + y_j & j = i \\ u_j & j \neq i, j \in \{1, 2, \dots, r_A + r_C\} \end{cases}. \quad (\text{B.46})$$

$z^* \in D$. $(y^*)^T (z^* - u) > 0$ contradicting (B.44). Therefore $y_i^* = 0$. For an $i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}$ if $u_i > 0$, assume $y_i^* > 0$, choose

$$z_j^* = \begin{cases} u_j + \alpha & j = i \\ u_j & j \neq i, j \in \{1, 2, \dots, r_A + r_C\} \end{cases} . \quad (\text{B.47})$$

where α is a positive real number. $z^* \in D$. $(y^*)^T (z^* - u) > 0$ contradicting (B.44).

Assume $y_i^* < 0$, choose

$$z_j^* = \begin{cases} \frac{u_j}{2} & j = i \\ u_j & j \neq i, j \in \{1, 2, \dots, r_A + r_C\} \end{cases} . \quad (\text{B.48})$$

$z^* \in D$. $(y^*)^T (z^* - u) > 0$ contradicting (B.44). Therefore $y_i^* = 0$. For an $i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}$ if $u_i = 0$ assume $y_i^* > 0$, choose

$$z_j^* = \begin{cases} y_i^* & j = i \\ u_j & j \neq i, j \in \{1, 2, \dots, r_A + r_C\} \end{cases} . \quad (\text{B.49})$$

$z^* \in D$. $(y^*)^T (z^* - u) > 0$ contradicting (B.44). Therefore $y_i^* \leq 0$. In summary

$$y_i^* = \begin{cases} 0 & i \in \{1, 2, \dots, r_A\} \\ 0 & i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}, u_i > 0 \\ \leq 0 & i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}, u_i = 0 \end{cases} . \quad (\text{B.50})$$

At this point I claim the following

$$y_i^* = \begin{cases} s_i & i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}, u_i = 0, s_i < 0 \\ 0 & i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}, u_i = 0, s_i \geq 0 \end{cases} . \quad (\text{B.51})$$

Notice that with this assignment (B.51),

$$(y^*)^T (z - u) = \sum_{i \in \{r_A+1, r_A+2, \dots, r_A+r_C\}, u_i=0, s_i < 0} s_i z_i \quad \forall z \in D. \quad (\text{B.52})$$

$$\leq 0 \text{ since } z_i \geq 0 \text{ from (B.2).} \quad (\text{B.53})$$

Hence y^* is a feasible point of (B.43) – (B.45). Assume that the assignment in (B.51) is incorrect. This means that there is another point x in $\mathbb{R}^{r_A+r_C}$ with

$$\|s - x\|^2 < \|s - y^*\|^2. \quad (\text{B.54})$$

Such a point x should have an $x_i > 0$ for an $i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}$, $u_i = 0$, $s_i \geq 0$. This would contradict the conditions given in the summary (B.50). Therefore,

$$y_i^* = \begin{cases} 0 & i \in \{1, 2, \dots, r_A\} \\ 0 & i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}, u_i > 0 \\ s_i & i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}, u_i = 0, s_i < 0 \\ 0 & i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}, u_i = 0, s_i \geq 0 \end{cases}. \quad (\text{B.55})$$

is the optimal solution of (B.43) – (B.45).

Q.E.D.

Claim 9. Let Ω be a non-empty, closed, convex set in $\mathbb{R}^n$. Let $P_\Omega(x)$ be the projection of $x \in \mathbb{R}^n$ on Ω .

$$(P_\Omega(x) - x)^T (x - y) \leq 0 \quad \forall y \in \Omega \quad (\text{B.56})$$

Proof. For $y \in \Omega$

$$\alpha P_\Omega(x) + (1 - \alpha)y \in \Omega \text{ for } \alpha \in (0, 1); \quad (\text{B.57})$$

since Ω is a convex set. Let

$$f_x(y) = \frac{1}{2} \|y - x\|^2 \text{ be a function from } \mathbb{R}^n \rightarrow \mathbb{R}. \quad (\text{B.58})$$

$$f_x(P_\Omega(x)) \leq f_x(\alpha P_\Omega(x) + (1 - \alpha)y) \quad \alpha \in (0, 1) \quad (\text{B.59})$$

from the definition of projection operator. (B.59) can be written as

$$\frac{1}{2} \|x - P_\Omega(x)\|^2 \leq \frac{1}{2} \|\alpha(P_\Omega(x) - x) + (1 - \alpha)(y - x)\|^2 \quad \alpha \in (0, 1). \quad (\text{B.60})$$

Expanding the right hand side of (B.60) we obtain:

$$\begin{aligned} \frac{1}{2} \|x - P_\Omega(x)\|^2 &\leq \frac{1}{2} \alpha^2 \|P_\Omega(x) - x\|^2 + \frac{1}{2} (1 - \alpha)^2 \|y - x\|^2 \\ &\quad + \alpha(1 - \alpha) \left((P_\Omega(x) - x)^T (y - x) \right) \quad \alpha \in (0, 1). \end{aligned} \quad (\text{B.61})$$

We can rearrange (B.61) as

$$\begin{aligned} \frac{1}{2} \|x - P_\Omega(x)\|^2 (1 - \alpha)(1 + \alpha) &\leq \frac{1}{2} (1 - \alpha)^2 \|y - x\|^2 \\ &\quad + \alpha(1 - \alpha) \left((P_\Omega(x) - x)^T (y - x) \right) \quad \alpha \in (0, 1). \end{aligned} \quad (\text{B.62})$$

Since $\alpha \in (0, 1)$ we can divide both sides of (B.62) by $(1 - \alpha)$.

$$\frac{1}{2} \|x - P_\Omega(x)\|^2 (1 + \alpha) \leq \frac{1}{2} (1 - \alpha) \|y - x\|^2 + \alpha \left((P_\Omega(x) - x)^T (y - x) \right) \quad \alpha \in (0, 1). \quad (\text{B.63})$$

Note that if $x = P_\Omega(x)$ then (B.56) holds. In the other case

$$\|x - P_\Omega(x)\|^2 > 0. \quad (\text{B.64})$$

Let $\alpha_k = (1 - \frac{1}{k})$ $k \geq 2$ be a sequence. Since $\alpha_k \in (0, 1)$ we can replace α_k with α in (B.63) and we can write

$$\left(1 - \frac{1}{2k}\right) \|x - P_\Omega(x)\|^2 \leq \frac{1}{2k} \|y - x\|^2 + \left(1 - \frac{1}{k}\right) (P_\Omega(x) - x)^T (y - x). \quad (\text{B.65})$$

Assume

$$(P_\Omega(x) - x)^T (y - x) < 0. \quad (\text{B.66})$$

(B.65) and (B.66) imply that for integer $k \geq \max(2, \frac{\|y-x\|^2}{(P_\Omega(x)-x)^T(y-x)})$

$$0 < \left(1 - \frac{1}{2k}\right) \|x - P_\Omega(x)\|^2 \leq (P_\Omega(x) - x)^T (y - x) < 0. \quad (\text{B.67})$$

This is a contradiction.

Q.E.D.

Claim 10. *Let Ω be a non-empty, closed, convex set in $\mathbb{R}^n$. For an $x \in \mathbb{R}^n$*

$$\|P_\Omega(x) - y\| \leq \|x - y\| \quad \forall y \in \Omega \quad (\text{B.68})$$

Proof.

$$\|P_\Omega(x) - y\|^2 - \|x - y\|^2 = (P_\Omega(x) - y)^T (P_\Omega(x) - y) - (x - y)^T (x - y) \quad \forall y \in \Omega \quad (\text{B.69})$$

$$= P_\Omega(x)^T P_\Omega(x) - 2P_\Omega(x)^T y - x^T x + 2x^T y \quad \forall y \in \Omega \quad (\text{B.70})$$

A fundamental property of the projection operator is [79]

$$(x - P_\Omega(x))^T (y - P_\Omega(x)) \leq 0 \quad \forall y \in \Omega \quad (\text{B.71})$$

When we expand expressions in (B.56) and in (B.71) we obtain

$$(P_{\Omega}(x) - x)^T (x - y) = P_{\Omega}(x)^T x - P_{\Omega}(x)^T y - x^T x + x^T y \leq 0 \quad \forall y \in \Omega \quad (\text{B.72})$$

$$(x - P_{\Omega}(x))^T (y - P_{\Omega}(x)) = x^T y - x^T P_{\Omega}(x) - P_{\Omega}(x)^T y + P_{\Omega}(x)^T P_{\Omega}(x) \leq 0 \quad \forall y \in \Omega. \quad (\text{B.73})$$

Summing side by side we obtain

$$2x^T y - 2P_{\Omega}(x)^T y + P_{\Omega}(x)^T P_{\Omega}(x) - x^T x \leq 0 \quad \forall y \in \Omega. \quad (\text{B.74})$$

(B.70) and (B.74) imply

$$\|P_{\Omega}(x) - y\|^2 - \|x - y\|^2 \leq 0 \quad \forall y \in \Omega. \quad (\text{B.75})$$

In the following a subgradient algorithm that generates a sequence $u^k \in D$ for $k \geq 0$ is described. It is proved that $\bar{\phi}(u^k) \rightarrow \phi(u^*)$ where $u^* \in D$ is a minimizing point of $\phi(u)$ in D . That minimum is assumed to exist. Here $\bar{\phi}(u^k) = \min_{0 \leq t \leq k} \phi(u^t)$. Definitions related to this sequence and the assumptions made are formally stated below.

1. u^* an optimal solution of

$$\min \quad \phi(u) \quad (\text{B.76})$$

s.t.

$$u \in D \quad (\text{B.77})$$

2.

$$u^0 \in D \quad (\text{B.78})$$

$$u^{k+1} = P_D(u^k - \lambda^k d^k) \quad k \geq 0 \quad (\text{B.79})$$

$$d^k = \begin{cases} c^k & \text{if } \exists i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}, u_i^k = 0 \text{ and} \\ & c_i^k + \delta^k d_i^{k-1} > 0 \\ c^k + \delta^k d^{k-1} & \text{otherwise} \end{cases} \quad (\text{B.80})$$

$$d^{-1} = 0 \quad (\text{B.81})$$

$$\delta^k = \begin{cases} -\frac{\tau^k (c^k)^T d^{k-1}}{\|d^{k-1}\|^2} & (c^k)^T d^{k-1} < 0 \\ 0 & \text{otherwise} \end{cases} \quad (\text{B.82})$$

$$c_i^k = \begin{cases} 0 & i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}, u_i^k = 0, s_i^k > 0 \\ s_i^k & \text{otherwise} \end{cases} \quad (\text{B.83})$$

$$\tau^k \geq 1 \quad (\text{B.84})$$

$$s^k \text{ is a subgradient at } u^k \quad (\text{B.85})$$

$$\{s^k, k \geq 0\} \text{ is a bounded set} \quad (\text{B.86})$$

By Claim 5 subgradient vector defined in (B.22) has this property.

$$\lambda^k \|d^k\|^2 \leq \phi(u^k) - \phi(u^*) \quad k \geq 0 \quad (\text{B.87})$$

$$\lambda^k > 0 \quad (\text{B.88})$$

$$\lim_{k \rightarrow \infty} \lambda^k = 0 \quad (\text{B.89})$$

$$\lim_{n \rightarrow \infty} \sum_{k=0}^n \lambda_k \text{ does not exist (increases without bound)} \quad (\text{B.90})$$

Notice that by its definition in (B.80), d^k is never an infeasible direction. Whenever the deflected direction is infeasible at the boundary of D , the generation procedure described above switches to the conditional subgradient c^k . Conditional subgradient

algorithm is due to the work presented in [81].

Claim 11. *For a $k \geq 0$ if $-c_i^{k+1} - \delta^{k+1}d_i^k$ at u^{k+1} is not an infeasible direction then*

$$(d^k)^T d^{k+1} \geq 0. \quad (\text{B.91})$$

Proof.

$$(d^k)^T d^{k+1} = (d^k)^T (c^{k+1} + \delta^{k+1}d^k) = (d^k)^T c^{k+1} + \delta^{k+1} (d^k)^T d^k \text{ from (B.80).} \quad (\text{B.92})$$

If $(d^k)^T c^{k+1} \geq 0$ then $\delta^{k+1} = 0$ from (B.82). This implies (B.91) together with (B.92).

If $(d^k)^T c^{k+1} < 0$ then

$$(d^k)^T c^{k+1} + \delta^{k+1} (d^k)^T d^k = (1 - \tau^{k+1}) (d^k)^T c^{k+1} \text{ from (B.82).} \quad (\text{B.93})$$

Because $\tau_{k+1} \geq 1$ from (B.84)

$$(1 - \tau^{k+1}) (d^k)^T c^{k+1} \geq 0. \quad (\text{B.94})$$

Q.E.D.

In a pure deflected subgradient algorithm that does not switch to conditional choosing τ_k as 1.5 is recommended in [82].

Claim 12. $\{d^k, k \geq 0\}$ defined in (B.80) is a bounded set.

Proof.

$$d^k \text{ is either } c^k - \frac{\tau^k (c^k)^T d^{k-1}}{\|d^{k-1}\|^2} d^{k-1} \text{ or simply } c^k \quad (\text{B.95})$$

from (B.80), (B.82). We know that $\{s^k, k \geq 0\}$ is a bounded set (B.86). With the definition of c^k (B.83) this implies that

$$\{c^k, k \geq 0\} \text{ is a bounded set.} \quad (\text{B.96})$$

$$\left\| \frac{-\tau^k (c^k)^T (d^{k-1})}{\|d^{k-1}\|^2} d^{k-1} \right\| = \left| \frac{-\tau^k (c^k)^T (d^{k-1})}{\|d^{k-1}\|} \right| \leq \tau^k \|c^k\| \left\| \frac{d^{k-1}}{\|d^{k-1}\|} \right\|$$

$$\text{from Cauchy-Schwarz inequality.} \quad (\text{B.97})$$

From (B.95), (B.96), (B.97) boundedness of $\{d^k, k \geq 0\}$ is deduced. Q.E.D.

Claim 13. *Let ε be a positive real number. If*

$$\phi(u^k) - \phi(u^*) \geq \varepsilon \quad \forall k \geq 0 \quad (\text{B.98})$$

then

$$(d^k)^T (u^* - u^k) \leq (c^k)^T (u^* - u^k) \quad \forall k \geq 0. \quad (\text{B.99})$$

Proof. If $k = 0$ then $d^0 = c^0$ from (B.80) and (B.81). Therefore (B.99) holds. Assume (B.99) holds for a $k \geq 0$. d^{k+1} is either equal to c^{k+1} or $c^{k+1} + \delta^{k+1} d^k$ from (B.80). In the former case

$$(d^{k+1})^T (u^* - u^{k+1}) = (c^{k+1})^T (u^* - u^{k+1}) \leq (c^{k+1})^T (u^* - u^{k+1}). \quad (\text{B.100})$$

In the latter case,

$$(d^{k+1})^T (u^* - u^{k+1}) = (c^{k+1} + \delta^{k+1} d^k)^T (u^* - u^{k+1}) \quad (\text{B.101})$$

$$= (c^{k+1})^T (u^* - u^{k+1}) + \delta^{k+1} (d^k)^T (u^* - u^{k+1}). \quad (\text{B.102})$$

Set

$$p^k = P_D(u^k - \lambda^k d^k) - (u^k - \lambda^k d^k). \quad (\text{B.103})$$

$$(d^k)^T (u^* - u^{k+1}) = (d^k)^T (u^* - P_D(u^k - \lambda^k d^k)) \text{ from (B.79)} \quad (\text{B.104})$$

$$= (d^k)^T (u^* - p^k - (u^k - \lambda^k d^k)) \text{ from (B.103)} \quad (\text{B.105})$$

$$= (d^k)^T (u^* - u^k) + \lambda^k \|d^k\|^2 - (d^k)^T p^k \quad (\text{B.106})$$

$$- (d^k)^T p^k \leq 0 \text{ from Claim 9 and (B.88)}. \quad (\text{B.107})$$

(B.106), (B.107) imply

$$(d^k)^T (u^* - u^{k+1}) \leq (d^k)^T (u^* - u^k) + \lambda^k \|d^k\|^2. \quad (\text{B.108})$$

$$(d^k)^T (u^* - u^k) \leq (c^k)^T (u^* - u^k) \quad (\text{B.109})$$

since we assumed that the claim is true for k . Given a subgradient vector s in $\mathbb{R}^{r_A+r_C}$ at point $u^k \in D$, consider the vector

$$c^k = s^k + P_{N_D(u^k)}(-s^k). \quad (\text{B.110})$$

c^k is also a subgradient vector at u^k from Claim 6. Using Claim 8, c^k can be written as

$$c_i^k = \begin{cases} 0 & i \in \{r_A + 1, r_A + 2, \dots, r_A + r_C\}, u_i^k = 0, s_i^k > 0 \\ s_i^k & \text{otherwise} \end{cases}. \quad (\text{B.111})$$

This c^k (B.111) is the vector defined in (B.83). From the definition of the subgradient vector (B.22)

$$(c^k)^T (u^* - u^k) \leq \phi(u^*) - \phi(u^k) \quad (\text{B.112})$$

$$\phi(u^*) - \phi(u^k) \leq -\lambda^k \|d^k\|^2 \quad (\text{B.113})$$

from (B.87), (B.109), (B.112), (B.113) imply

$$(d^k)^T (u^* - u^k) \leq -\lambda^k \|d^k\|^2. \quad (\text{B.114})$$

(B.108), (B.114) imply

$$(d^k)^T (u^* - u^{k+1}) \leq 0. \quad (\text{B.115})$$

Nonnegativity of δ^{k+1} , (B.115), (B.102) imply

$$(d^{k+1})^T (u^* - u^{k+1}) \leq (c^{k+1})^T (u^* - u^{k+1}). \quad (\text{B.116})$$

Proposition 29. *Under the above stated assumptions*

$$\bar{\phi}(u^k) \rightarrow \phi(u^*). \quad (\text{B.117})$$

where $\bar{\phi}(u^k) = \min_{0 \leq t \leq k} \phi(u^t)$.

Proof. Assume (B.117) is false. This implies

$$\exists \varepsilon > 0 \ \forall N \geq 0 \ \exists k \geq N \quad |\bar{\phi}(u^k) - \phi(u^*)| \geq \varepsilon \quad (\text{B.118})$$

By definition of u^* in (B.76)–(B.77) we can remove the absolute value sign in (B.118) :

$$\exists \varepsilon > 0 \ \forall N \geq 0 \ \exists k \geq N \ \bar{\phi}(u^k) - \phi(u^*) \geq \varepsilon \quad (\text{B.119})$$

Since $\bar{\phi}(u^k)$ is a non-increasing sequence we can conclude:

$$\phi(u^k) - \phi(u^*) \geq \bar{\phi}_k - \phi(u^*) \geq \varepsilon \quad \forall k \geq 0. \quad (\text{B.120})$$

From the definition of u^{k+1} given in (B.79)

$$\|u^{k+1} - u^*\|^2 = \|P_D(u^k - \lambda^k d^k) - u^*\|^2 \quad \forall k \geq 0 \quad (\text{B.121})$$

$$\leq \|u^k - \lambda^k d^k - u^*\|^2 \quad \forall k \geq 0 \text{ from Claim 10} \quad (\text{B.122})$$

$$\leq \|u^k - u^*\|^2 - 2\lambda^k (d^k)^T (u^k - u^*) + (\lambda^k)^2 \|d^k\|^2 \quad \forall k \geq 0 \text{ (we simply expand (B.122))} \quad (\text{B.123})$$

Since c^k is a subgradient at u^k we can write from (B.22),

$$\phi(u^*) \geq \phi(u^k) + (u^* - u^k)^T c^k \quad \forall k \geq 0. \quad (\text{B.124})$$

Due to (B.120) assumptions of Claim 13 are satisfied. Consequently,

$$(d^k)^T (u^* - u^k) \leq (c^k)^T (u^* - u^k) \quad \forall k \geq 0. \quad (\text{B.125})$$

(B.120), (B.124), (B.125), (B.88) imply

$$-2\lambda^k (d^k)^T (u^k - u^*) \leq -2\lambda^k \varepsilon \quad \forall k \geq 0. \quad (\text{B.126})$$

(B.89) and Claim 12 imply

$$\exists K_1 \geq 0 \text{ such that } \forall k \geq K_1 \ \lambda^k \|d^k\|^2 \leq \varepsilon. \quad (\text{B.127})$$

(B.123), (B.126), (B.127) imply

$$\|u^{k+1} - u^*\|^2 \leq \|u^k - u^*\|^2 - \lambda^k \varepsilon \quad \forall k \geq K_1. \quad (\text{B.128})$$

Writing (B.128) for $K_1 \leq k \leq N$ with $N > K_1$, summing side by side we obtain

$$\|u^N - u^*\|^2 \leq \|u^{K_1} - u^*\|^2 - (\lambda^{K_1} + \lambda^{K_1+1} + \dots + \lambda^{N-1}) \varepsilon. \quad (\text{B.129})$$

As $N \rightarrow \infty$, from (B.90) right hand side of (B.129) decreases without bound. Contradiction. Q.E.D.

REFERENCES

1. Akyıldız, I. F., T. Melodia and K.R. Chowdhury, "A Survey on Wireless Multimedia Sensor Networks", *Computer Networks*, Vol. 51, No. 4, pp. 921-960, 2007.
2. Yick, J., B. Mukherjee and D. Ghosal, "Wireless Sensor Network Survey", *Computer Networks*, Vol. 52, pp. 2292-2330, 2008.
3. Cardei, M. and J. Wu, *Coverage in Wireless Sensor Networks*, Technical Report, Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, Florida, 2006.
4. Wang, L. and Y. Xiao, "A Survey of Energy-efficient Scheduling Mechanisms in Sensor Networks", *Mobile Networks and Applications*, Vol. 11, No. 5, pp. 723-740, October 2006.
5. Akkaya K., M. Younis, "A Survey on Routing Protocols for Wireless Sensor Networks", *Ad Hoc Networks*, Vol. 3, No. 3, pp. 325-349, 2005.
6. Oliveira, C. A. S. and P. M. Pardalos, "A Survey of Combinatorial Optimization Problems in Multicast Routing", *Computers and Operations Research*, Vol. 32, No. 8, pp. 1953-1981, 2005.
7. Bogdanov A., E. Maneva and S. Riesenfeld, "Power-aware Base Station Positioning for Sensor Networks", *Proceedings of the IEEE Computer and Communications Conference*, pp. 575-585, 2004.
8. Dhillon, S. S. and K. Chakrabarty, "Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks", *Proceedings of the IEEE Wireless Communications and Networking Conference*, pp. 1609-1614, 2003.
9. Chakrabarty K., S.S. Iyengar, H. Qi and E. Cho, "Grid Coverage for Surveillance

- and Target Location in Distributed Sensor Networks”, *IEEE Transactions on Computers*, Vol. 51, No. 12, pp. 1448-1453, 2002.
10. Yang, S., F. Dai, M. Cardei, J. Wu and F. Patterson, “On Connected Multiple Point Coverage in Wireless Sensor Networks”, *International Journal of Wireless Information Networks*, Vol. 13, No. 4, pp. 289-301, 2006.
 11. Altmel, İ. K., N. Aras, E. Güney and C. Ersoy, “Effective Coverage in Sensor Networks: Binary Integer Programming Formulations and Heuristics”, *Proceedings of IEEE International Conference on Communications*, pp. 4014-4019, 2006.
 12. Altmel, İ. K., N. Aras, E. Güney and C. Ersoy, “Binary Integer Programming Formulation and Heuristics for Differentiated Coverage in Heterogeneous Sensor Networks”, *Computer Networks*, Vol. 52, No. 12, pp. 2419-2431, 2008.
 13. Wang, J. and N. Zhong, “Efficient Point Coverage in Wireless Sensor Networks”, *Journal of Combinatorial Optimization*, Vol. 11, No. 3, pp. 291-304, 2006.
 14. Ganesan, D., R. Cristescu and B. Beferull-Lozano, “Power Efficient Sensor Placement and Transmission Structure for Data Gathering under Distortion Constraints”, *Proceedings of Third International Symposium on Information Processing in Sensor Networks*, pp. 142-150, 2004.
 15. Osais, Y. E., M. Hilaire and F. R. Yu, “Directional Sensor Placement with Optimal Sensing Range, Field of View and Orientation”, *Mobile Networks and Applications*, Vol. 15, No. 2, pp. 216-225, 2010.
 16. Nakamura, F. G., F. P. Quintao, G. C. Menezes and G. R. Mateus, “An Optimal Node Scheduling for Flat Wireless Sensor Networks”, *Lecture Notes in Computer Science*, Vol. 3420, pp. 475-482, 2005. *Comput. Sc.* 3420 (2005) 475–482.
 17. Cardei, M., M. T. Thai, Y. Li and W. Wu, “Energy-efficient Target Coverage in Wireless Sensor Networks”, *Proceedings of IEEE Computer and Communications*

Societies Conference, pp. 1976-1984, 2005.

18. Nanez, E. C., A. Garcia and C. Li, "A Game-theoretic Approach to Efficient Power Management in Sensor Networks", *Operations Research*, Vol. 56, pp. 552-561, 2008.
19. Alfieri, A., A. Bianco, P. Brandimarte and C. F. Chiasserini, "Maximizing System Lifetime in Wireless Sensor Networks", *European Journal of Operational Research*, Vol. 181, pp. 390-402, 2007.
20. Ha, R. W., P. Ho, X. S. Shen and J. Zhang, "Sleep Scheduling for Wireless Sensor Networks via Network Flow Model", *Computer Communications*, Vol. 29, No. 13-14, pp. 2469-2481, 2006.
21. Liu, H., P. Wan and X. Jia, "Maximal Lifetime Scheduling for K to 1 Sensor-target Surveillance Networks", *Computer Networks*, Vol. 50, No. 15, pp. 2839-2854, 2006.
22. Zhao, Q. and M. Gurusamy, "Connected K-target Coverage Problem in Wireless Sensor Networks", *Computer Networks*, Vol. 52, No. 11, pp. 2205-2220, 2008.
23. Ai, J. and A. A. Abouzeid, "Coverage by Directional Sensors in Randomly Deployed Wireless Sensor Networks", *Journal of Combinatorial Optimization*, Vol. 11, pp. 21-41, 2006.
24. Yan, T., T. He and J. A. Stankovic, "Differentiated Surveillance for Sensor Networks", *Proceedings of the First International Conference on Embedded Networked Sensor Systems*, pp. 51-62, 2003.
25. Yardibi, T. and E. Karasan, "A Distributed Activity Scheduling Algorithm for Wireless Sensor Networks With Partial Coverage", *Wireless Networks*, Vol. 16, No. 1, pp. 213-225, 2010.
26. Younis, M., M. Youssef and K. Arisha, "Energy-aware Routing in Cluster-based Sensor Networks", *Proceedings of 10th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, pp. 129-

136, 2002.

27. Sankar, A. and Z. Liu, "Maximum Lifetime Routing in Wireless Ad-hoc Networks", *Proceedings of Twenty-Third Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 1089-1097, 2004.
28. Chang, J.-H. and L. Tassiulas, "Maximum Lifetime Routing in Wireless Sensor Networks", *IEEE Transactions on Networking*, Vol. 12, pp. 609-619, 2004.
29. Xue, Y., Y. Cui and K. Nahrstedt, "Maximizing Lifetime for Data Aggregation in Wireless Sensor Networks", *Mobile Networks and Applications*, Vol. 10, No. 6, pp. 853-864, 2005.
30. Hua, C. and T. P. Yum, "Maximum Lifetime Routing and Data Aggregation for Wireless Sensor Networks", *Lecture Notes in Computer Science*, Vol. 3976, pp. 840-855, 2006.
31. Ergen, S. C. and P. Varaiya, "Energy Efficient Routing with Delay Guarantee for Sensor Networks", *Wireless Networks*, Vol. 13, No. 5, pp. 679-690, 2007.
32. Pham, M., D. Kim, T. Kim and S. Yoo, "Maximize the Coverage Lifetime of Sensor Networks", *Lecture Notes in Computer Science*, Vol. 4097, pp. 475-484, 2006.
33. Ciciriello, P., L. Mottola and G. P. Picco, "Efficient Routing from Multiple Sources to Multiple Sinks in Wireless Sensor Networks", *Lecture Notes in Computer Science*, Vol. 4373, pp. 35-40, 2007.
34. Cui, Y., Y. Xue and K. Nahrstedt, "A Utility-based Distributed Maximum Lifetime Routing Algorithm for Wireless Networks", *IEEE Transactions on Vehicular Technology*, Vol.55, No. 3, 2006.
35. Norman, J., Joseph J. P. and Roja P. P., "A Faster Routing Scheme For Stationary Wireless Sensor Networks-A Hybrid Approach", *International Journal of Ad hoc, Sensor and Ubiquitous Computing*, Vol. 1, No.1, 2010.

36. Vincze, Z., R. Vida and A. Vidacs, "Deploying Multiple Sinks in Multi-hop Wireless Sensor Networks", *Proceedings of IEEE International Conference on Pervasive Services*, pp. 55-63, 2007.
37. Basagni, S., A. Carosi, E. Melachrinoudis, C. Petrioli and Z. M. Wang, "Controlled Sink Mobility for Prolonging Wireless Sensor Networks Lifetime", *Wireless Networks*, Vol. 14, pp. 831-858, 2008.
38. Patel, M., R. Chandrasekaran and S. Venkatesan, "Energy Efficient Sensor, Relay and Base Station Placements for Coverage, Connectivity and Routing", *Proceedings of 24th IEEE International Performance, Computing, and Communications Conference*, pp. 581-586, 2005.
39. Hou, Y. T., Y. Shi, H. D. Sherali and S. F. Midkiff, "On Energy Provisioning and Relay Node Placement for Wireless Sensor Networks", *IEEE Transactions on Wireless Communications*, Vol. 4, No. 5, pp. 2579-2590, 2005.
40. Gatzianas, M. and L. Georgiadis, "A Distributed Algorithm for Maximum Lifetime Routing in Sensor Networks with Mobile Sink", *IEEE Transactions on Wireless Communications*, Vol. 7, pp. 984-994, 2008.
41. Efrat, A., S. Har-Peled and J. Mitchell, "Approximation Algorithms for two Optimal Location Problems in Sensor Networks", *Proceedings of 2nd International Conference on Broadband Networks*, pp. 767-776, 2005.
42. Shi, Y. and Y. T. Hou, "Approximation Algorithm for Base Station Placement in Wireless Sensor Networks", *Proceedings of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pp. 512-519, 2007.
43. Shi, Y., Y. T. Hou and A. Efrat, "Algorithm Design for a Class of Base Station Location Problems in Sensor Networks", *Wireless Networks*, Vol. 15, pp. 21-28, 2009.

44. Kim, H., Y. Seok, N. Choi, Y. Choi and T. Kwon, "Optimal Multi-sink Positioning and Energy-efficient Routing in Wireless Sensor Networks", *Lecture Notes in Computer Science*, Vol. 3391, pp. 264-274, 2005.
45. Türkoğulları, Y. B., N. Aras, İ. K. Altinel and C. Ersoy, "Optimal Placement and Activity Scheduling to Maximize Coverage Lifetime in Wireless Sensor Networks", *Proceedings of the 22nd International Symposium on Computer and Information Sciences*, pp. 1-6, 2007.
46. <http://www.xbow.com/Support/wUserManuals.aspx>.
47. Garey, M. R. and D. S. Johnson, 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman.
48. Balas, E., "Cutting Planes from Conditional Bounds: A New Approach to Set Covering", *Mathematical Programming Study*, Vol. 12, pp. 19-36, 1980.
49. Balas, E. and A. Ho, "Set Covering Algorithms Using Cutting Planes, Heuristics, and Subgradient Optimization: A Computational Study", *Mathematical Programming*, Vol. 12, pp. 37-60, 1980.
50. Balas, E. and S. M. NG, "On the Set Covering Polytope: I. All the Facets with Coefficients in $\{0, 1, 2\}$ ", *Mathematical Programming*, Vol. 43, pp. 57-69, 1989.
51. Balas, E. and S. M. NG, "On the Set Covering Polytope: II. Lifting the Facets with Coefficients in $\{0, 1, 2\}$ ", *Mathematical Programming*, Vol. 45, pp. 1-20, 1989.
52. Cornuéjols, G. and A. Sassano, "On the 0,1 Facets of the Set Covering Polytope", *Mathematical Programming*, Vol. 43, pp. 45-55, 1989.
53. Nobili, P. and A. Sassano, "Facets and Lifting Procedures for the Set Covering Polytope", *Mathematical Programming*, Vol. 45, pp. 111-137, 1989.
54. Sassano, A., "On the Facial Structure of the Set Covering Polytope", *Mathematical*

- Programming*, Vol. 44, pp. 181-202, 1989.
55. Sánchez, M., M. I. Sobrón and B. Vitoriano, “On the Set Covering Polytope: Facets with Coefficients in $\{0, 1, 2, 3\}$ ”, *Annals of Operations Research*, Vol. 81, pp. 343-356, 1998.
 56. Saxena, A., “On the Set Covering Polytope: I. All Facets with Coefficients in $\{0, 1, 2, 3\}$ ”, *GSIA Working Paper #2004-E29*, 2004.
 57. Saxena, A., “On the Set Covering Polytope: II. Lifting Facets with Coefficients in $\{0, 1, 2, 3\}$ ”, *GSIA Working Paper #2004-E30*, 2004.
 58. Nemhauser, G. L. and L. A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, 1988.
 59. Schrijver, A., *Combinatorial Optimization Polyhedra and Efficiency Volume C*, chapters 77-80, Springer, 2003.
 60. Karmarkar, N., “A New Polynomial-time Algorithm for Linear Programming”, *Combinatorica*, Vol. 4, pp. 373-395, 1984.
 61. Duffin, R. J., “Infinite Programs”, in H. W. Kuhn and A. W. Tucker (eds.), *Linear Inequalities and Related Systems*, pp. 157-170, Princeton University Press, Princeton, NJ, 1956.
 62. <http://www.ilog.com>.
 63. Zhang, H. and J. Hou, “Maintaining Sensing Coverage and Connectivity in Large Sensor Networks”, *International Journal of Wireless Ad Hoc and Sensor Networks*, Vol. 1, pp. 89-124, 2005.
 64. Wentges, P., “Weighted Dantzig-Wolfe Decomposition for Linear Mixed-integer Programming”, *International Transactions in Operational Research*, Vol. 4, pp. 151-162, 1997.

65. Klose, A. and S. Görtz, “A Branch-and-price Algorithm for the Capacitated Facility Location Problem”, *European Journal of Operational Research*, Vol. 179, pp. 1109-1125, 2007.
66. Guta, B., *Subgradient Optimization Methods in Integer Programming with an Application to a Radiation Therapy Problem*, Ph.D. Thesis, University of Kaiserslautern, 2003.
67. Barnhart, C., E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh and P. H. Vance, “Branch-and-Price: Column Generation for Solving Huge Integer Programs”, *Operations Research*, Vol. 46, pp. 316-329, 1998.
68. Türkoğulları, Y. B., N. Aras, İ. K. Altinel and C. Ersoy, “Optimal Placement, Scheduling and Routing to Maximize Lifetime in Sensor Networks”, *Journal of the Operational Research Society*, Vol. 61, No.6, pp. 1000-1012, 2010.
69. Beasley, J., “Lagrangian Relaxation”, in C. Reeves (eds.), *Modern Heuristic Techniques for Combinatorial Problems*, pp. 243-303, John Wiley and Sons, New York, 1993.
70. Türkoğulları, Y. B., N. Aras, İ. K. Altinel and C. Ersoy, “A Column Generation Based Heuristic for Sensor Placement, Activity Scheduling and Data Routing in Wireless Sensor Networks”, *European Journal of Operations Research*, accepted for publication, 2010.
71. Desaulniers, G., J. Desrosiers and M. Solomon, *Column Generation*, Springer, 2005.
72. Dantzig, G. B. and P. Wolfe, “Decomposition Principle for Linear Programs”, *Operations Research*, Vol. 8, pp. 101-111, 1960.
73. Bazaraa, M. S., J. J. Jarvis and H. D. Sherali, *Linear Programming and Network Flows*, 4 th Edition, Wiley, 2010.
74. Sundararaman, B., U. Buy and A. D. Kshemkalyani, “Clock Synchronization for

- Wireless Sensor Networks: A Survey”, *Ad Hoc Networks*, Vol. 3, pp. 281-323, 2005.
75. Stathopoulos, T., L. Girod, J. Heidemann and D. Estrin, *Centralized Routing for Resource-constrained Wireless Sensor Networks*, Center for Embedded Networked Sensing Research Review, 2006.
 76. Türkoğulları, Y. B., N. Aras, İ. K. Altinel and C. Ersoy, “An Efficient Heuristic for Placement, Scheduling and Routing in Wireless Sensor Networks”, *Ad Hoc Networks*, Vol.8, No. 6, pp. 654-667, 2010.
 77. Torres, M. G. C. and J. Kabara, “Measuring Energy Consumption in Wireless Sensor Networks Using GSP”, *Proceedings of the 17th International Symposium on Personal, Indoor and Mobile Radio Communications*, 2006.
 78. <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/scpinfo.html>.
 79. Hiriart-Urruty, J. B. and C. Lemarechal, *Convex Analysis and Minimization Algorithms I*, Springer-Verlag, 1993.
 80. Bertsekas, D., *Nonlinear Programming*, Athena Scientific, 2002.
 81. Larsson, T., M. Patriksson and A. B. Stromberg, “Conditional Subgradient Optimization: Theory and Applications”, *European Journal of Operations Research*, Vol. 88, pp. 382-403, 1996.
 82. Camerini, P. M., L. Fratta and F. Maffioli, “On Improving Relaxation Methods by Modified Gradient Techniques”, *Mathematical Programming Study*, Vol. 3, pp. 26-34, 1975.