University of Exeter
Department of Computer Science

Hybrid Evolutionary Routing Optimisation for Wireless Sensor Mesh Networks

Alma As-Aad Mohammad Rahat

December, 2015

Supervised by Professor Richard Everson & Dr Jonathan Fieldsend

Submitted by Alma As-Aad Mohammad Rahat, to the University of Exeter as a thesis for the degree of Doctor of Philosophy in Computer Science , December, 2015.

This thesis is available for Library use on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

I certify that all material in this thesis which is not my own work has been identified and that no material has previously been submitted and approved for the award of a degree by this or any other University.

(signature) .....

Abstract

Battery powered wireless sensors are widely used in industrial and regulatory monitoring applications. This is primarily due to the ease of installation and the ability to monitor areas that are difficult to access. Additionally, they can be left unattended for long periods of time. However, there are many challenges to successful deployments of wireless sensor networks (WSNs). In this thesis we draw attention to two major challenges. Firstly, with a view to extending network range, modern WSNs use mesh network topologies, where data is sent either directly or by relaying data from node-to-node en route to the central base station. The additional load of relaying other nodes' data is expensive in terms of energy consumption, and depending on the routes taken some nodes may be heavily loaded. Hence, it is crucial to locate routes that achieve energy efficiency in the network and extend the time before the first node exhausts its battery, thus improving the network lifetime. Secondly, WSNs operate in a dynamic radio environment. With changing conditions, such as modified buildings or the passage of people, links may fail and data will be lost as a consequence. Therefore in addition to finding energy efficient routes, it is important to locate *combinations* of routes that are robust to the failure of radio links.

Dealing with these challenges presents a routing optimisation problem with multiple objectives: find good routes to ensure energy efficiency, extend network lifetime and improve robustness. This is however an NP-hard problem, and thus polynomial time algorithms to solve this problem are unavailable. Therefore we propose hybrid evolutionary approaches to approximate the optimal trade-offs between these objectives.

In our approach, we use novel search space pruning methods for network graphs, based on k-shortest paths, partially and edge disjoint paths, and graph reduction to combat the combinatorial explosion in search space size and consequently conduct rapid optimisation. The proposed methods can successfully approximate optimal Pareto fronts. The estimated fronts contain a wide range of robust and energy efficient routes. The fronts typically also include solutions with a network lifetime close to the optimal lifetime if the number of routes per nodes were unconstrained. These methods are demonstrated in a real network deployed at the Victoria & Albert Museum, London, UK.

for my family, friends, and mentors

Acknowledgements

It has been a great experience doing my PhD at the University of Exeter. I have learned more than I could ever have anticipated, and grown as a researcher. Here I would like to acknowledge those without whom this work would not have been possible.

First of all, I would like to thank my family for their unwavering support, and being a source of strength. I am indebted to them for all their sacrifices, and encouraging inquisitiveness and optimism. Without them I would not be the person I am today.

I am forever grateful to my supervisors, Richard Everson and Jonathan Fieldsend, for this once in a lifetime opportunity. Their knowledge, encouragement, enthusiasm, and expert supervision were pivotal in the completion of this thesis. I truly cherish all of our exhilarating discussions throughout my PhD, and I thank them for all their guidance, patience and time. It has been an honour and a privilege to work with them.

I would like to thank Mahesan Niranjan and Mark Nixon for introducing the exciting world of research to me, and being a source of inspiration. I would also like to thank Christopher Ferro for being my mentor throughout my PhD. I am grateful to Andrew Hinks for my first job in engineering, and being a mentor. I am also thankful to Rokeya Begum and Abul Hasan for instilling in me an interest of Physics and Mathematics while I was in school.

I would like to take this opportunity to thank Malik Hamid (and family), Mahbub Mishu, Meer Hossain, Faria Chowdhury, Wasifa Jamal, Clarissa Perks, Matthew Heritage, Calleb Karegyesa, Prashanth Srinivas, Jonathan Comley, David Walker, Jacqueline Christmas, Philip Sansom, Qiong Cao, Imran Sumel, Mainul Islam, Shafkat Chowdhury, Shabab Choudhury, Jonathan Wilkinson, Tanbirul Hasan, Vaibhav Shah, Raya Seerden, Jonathan Tsoi and Matthew Bain for their friendship and support.

Part of this work was supported by a *knowledge transfer partnership* (KTP) awarded to the IMC Group Ltd. and the University of Exeter (KTP008748). I would like to thank Martin Hancock (Technical Director, The IMC Group Ltd.), Neil Lundy (Engineering Manager, The IMC Group Ltd.), and Andrew Cleeves (Engineer, The IMC Group Ltd.) for their support and contributions. I would also like to thank Boris Pretzel (Chief Scientist, Victoria & Albert Museum), and Bhavesh Shah (Scientist, Victoria & Albert Museum), for facilitating the real network test.

I would like to thank the College of Engineering, Mathematics and Physical Sciences, University of Exeter, for financial support during my PhD.

Finally, I am grateful to Dragan Savić and Jürgen Branke for agreeing to examine this thesis.

Contents

List of Figures

Nomenclature and Abbreviations

Acronyms and abbreviations

BOLP	Bi-Objective Linear Program *(p. 80)*
DARPA	Defence Advanced Research Projects Agency *(p. 6)*
DLL	Data Link Layer *(p. 2)*
DSN	Distributed Sensor Networks *(p. 6)*
DSSS	Direct Sequence Spread Spectrum *(p. 8)*
EA	Evolutionary Algorithms *(p. 23)*
EAS	Empirical Attainment Surface *(p. 26)*
FEC	Forward Error Correction *(p. 2)*
FHSS	Frequency Hopping Spread Spectrum *(p. 8)*
HypE	Hypervolume Estimation algorithm for multi-objective optimisation *(p. 24)*
IBEA	Indicator Based Evolutionary Algorithm *(p. 24)*
IEEE	Institute of Electrical and Electronics Engineers *(p. 8)*
IETF	Internet Engineering Task Force *(p. 8)*
ISA	International Society for Automation *(p. 8)*
KSP	k-Shortest Paths *(p. 12)*
LP	Linear Program *(p. 4)*
LRWPAN	Low-Rate Wireless Personal Area Network *(p. 9)*
MAC	Medium Access Control *(p. 2)*
MOEA	Multi-objective Optimisation using Evolutionary Algorithms *(p. 23)*
MOEA/D	Multi-Objective Evolutionary Algorithm based on Decomposition *(p. 24)*
MOP	Multi-objective Optimisation Problem *(p. 21)*
MSOPS	Multiple Single Objective Pareto Sampling *(p. 24)*
NSGA	Non-dominated Sorting Genetic Algorithms *(p. 23)*
SAS	Summary Attainment Surface *(p. 25)*
SOLP	Single Objective Linear Program *(p. 80)*
TDMA	Time Division Multiple Access *(p. 2)*
WSN	Wireless Sensor Network *(p. 1)*

Symbols

A	Non-dominated archive: approximation of Pareto set *(p. 39)*
A_{ij}	Reception energy cost at node v_j for receiving data from node v_i *(p. 32)*
c	Crossover rate *(p. 39)*
C^k	Total energy cost at node v_k for a routing scheme $\mathcal{R}$ *(p. 32)*
C^k_{id}	Energy cost at node v_k in route R_{id} *(p. 60)*
Δ_{th}	Sidetrack cost at edge e_{th} *(p. 18)*
E	Set of edges in graph G *(p. 13)*
e_{ij}	Edge from v_i to v_j *(p. 13)*
$\mathcal{F}$	Objective vector *(p. 21)*
F	Fragility across the network for a routing scheme $\mathcal{R}$ *(p. 63)*
F_{id}	Fragility of a route $R_{id} \in \mathcal{R}$ *(p. 63)*
F_i	Fragility across all routes for node v_i *(p. 63)*
G	Network graph *(p. 13)*
G'	Reduced network graph using the resulting edge utilisations from solving maximum lifetime routing LP *(p. 37)*
G_{ij}	Adjacency matrix cell value corresponding to i-th row (node v_i) and j-th column (node v_j) *(p. 14)*
H_i	Total route cost for route R_i *(p. 13)*
HV	Hypervolume indicator *(p. 25)*
$\mathcal{I}_k$	Set of indices for nodes that transmit data to node v_k *(p. 32)*
λ_i	Relative importance of $f_i(x)$ in a weighted sum *(p. 23)*
$\mathcal{L}$	Network lifetime: the time before any node exhausts its battery *(p. 35)*
L'_k	Reciprocal of lifetime for node v_k due to routing scheme $\mathcal{R}$ *(p. 61)*
L_k	Lifetime of node v_k for routing scheme $\mathcal{R}$ *(p. 32)*
$\mathbf{M}$	Matrix of charge decay rates at nodes *(p. 50)*
μ	Perturbation rate *(p. 40)*
M	Number of edges in a network *(p. 13)*
N	Number of nodes in a network *(p. 13)*
N_c	Number of reporting cycles per unit time (e.g. one year) *(p. 32)*
$\mathcal{O}$	Asymptotic upper bound *(p. 16)*
O_k	Set of indices for nodes that receive data from v_k *(p. 32)*
π_m	Failure probability at edge $e_m \in E$ *(p. 62)*
$\mathcal{P}$	Path library *(p. 39)*
$\mathcal{P}_k$	Path library with k-shortest paths *(p. 37)*
PF	Pareto front *(p. 23)*
PS	Pareto set *(p. 22)*
$\tilde{p}_{id}$	All higher order product terms of a route failure probability p_{id} *(p. 62)*
q_i	Battery charge at node v_i *(p. 32)*
q_{0i}	Initial battery charge at node v_i *(p. 49)*
$\mathcal{R}$	A routing scheme: a set of routes where each node has one or more routes

	to the bases station *(p. 31)*
$R_{i[p]}$	p-th element of route R_i *(p. 13)*
R_i	A route from v_i to v_B *(p. 13)*
$\mathcal{T}$	Set of time shares for all routes $R_{id} \in \mathcal{R}$ *(p. 60)*
τ_{id}	Time share for d-th route from node v_i to the base station v_B *(p. 59)*
Θ	Asymptotic equivalence *(p. 14)*
T_{ij}	Transmission energy cost at node v_i for sending data to node v_j *(p. 32)*
$\mathcal{U}$	Set of priority nodes *(p. 33)*
U_i	Amount of data (flow) generated at node v_i *(p. 34)*
u_{ij}	Edge utilisation: the *(p. 32)*
V	Set of nodes in graph G *(p. 13)*
v_B	The base station *(p. 13)*
v_i	A node with identifier i *(p. 13)*
Ω	Search space after applying k-shortest path pruning in full graph G *(p. 37)*
Ω'	Search space after applying k-shortest path pruning in reduced graph G' *(p. 38)*
ω_{ij}	Composite edge cost for edge e_{ij} *(p. 37)*
w_{ij}	Edge cost from v_i to v_j *(p. 13)*
w_{ij}	Energy edge cost to transmit data for edge e_{ij} *(p. 31)*
$\mathcal{X}$	Decision space *(p. 21)*
x	An n-dimensional decision vector *(p. 21)*
$\mathcal{Y}$	Objective space *(p. 21)*

Publications

The materials presented in chapter 3 have been published in:

Rahat, A. A. M., Everson, R. M., and Fieldsend, J. E. (2014). Multi-objective routing optimisation for battery-powered wireless sensor mesh networks. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 1175–1182, New York, NY, USA. ACM. [**Best Paper in Real World Application Track**]

Rahat, A. A. M., Everson, R. M., and Fieldsend, J. E. (2015). Hybrid Evolutionary Approaches to Maximum Lifetime Routing and Energy Efficiency in Sensor Mesh Networks. *Evolutionary Computation*, 23(3):481–507.

Some of the materials presented in chapter 4 have been submitted for publication:

Rahat, A. A. M., Everson, R. M., and Fieldsend, J. E. (2015). Evolutionary Multi-Path Routing for Network Lifetime and Robustness in Wireless Sensor Networks. *Submitted for publication in Ad Hoc Networks.*

Chapter 1

Introduction

Wireless sensors are autonomous devices that measure environmental parameters such as temperature and humidity (Figure 1.1). In sensor networks, many such devices are distributed over a wide area. Generally, these sensors report data back to a central base station for further analyses from control or monitoring perspective. They are widely used in remote monitoring applications due to ease of installation and the ability to monitor areas that are difficult to access. Inevitably, such applications require these sensors to be battery powered.

In many commercial *Wireless Sensor Networks (WSNs)*, sensor nodes communicate directly to the base station in an ad hoc point-to-multi-point network topology (the direction of transmission is multi-point-to-point). Despite being a simple low power solution, such systems suffer from limited network range as all nodes must be able to communicate directly with the base station. Additionally, because of the strict network structure these networks are unable to cope with the changing conditions of the dynamic radio environment.

Figure 1.1 Typical wireless sensors. These standalone devices are designed to measure temperature and humidity. (©IMC Group Ltd.)

These shortcomings may be alleviated by using *mesh network* topologies, where data is relayed from node to node en route to the base station. This topology can extend the network range as nodes no longer need to be within direct radio range of the base station, and communication may be established with distant nodes as long as neighbouring nodes of that particular node can communicate with the base station (directly or indirectly). Furthermore, this topology permits the use of alternative routes in case an active route ceases to function due to link

or node failure.

Alongside powering the sensors, the major drain on batteries is the reception and transmission costs. In point-to-multi-point networks nodes only send their own data. Hence, the transmission and reception costs only depend upon parameters related to single direct links, e.g. baud rate (that is, data transmission rate) and transmission power. Therefore, the choice of routes, and thus the design of efficient networks is limited to selecting better baud rate and power configuration combinations, at least from an energy consumption perspective; generally, high baud rate and low power transmissions are preferred. In mesh networks, nodes have the additional responsibility of relaying other nodes' data. Depending on the paths taken by different nodes in the system, some nodes may be heavily loaded and as a consequence exhaust their batteries quickly. So it is desirable to locate good routes that ensure energy efficiency in a mesh network.

Minimising network energy consumption is equivalent to finding routes that maximise the *average lifetime* of the network [Rodoplu and Meng, 1999]. However, as Chang and Tassiulas [2004] indicated, using minimum energy routes throughout the lifetime of the network can be detrimental to a group of nodes that relay the most traffic. It is often more important to optimise routes with a view to maximising the time before any node exhausts its battery; this is the *network lifetime* – the time before the network first needs intervention to change a battery.

In WSNs, in addition to energy efficiency and network lifetime, it is important that data from the sensors are delivered reliably. Data delivery may fail for various reasons: node failure, congestion and link failure [Paradis and Han, 2007]. Nodes may fail because of hardware failure. In modern deployments, this is considered as a rare event due to recent advances in WSN hardware technology. Moreover, it is considered to be a critical event that requires immediate attention from the network administrator so that the faulty nodes may be replaced. Additionally, congestion may occur when multiple transmissions overlap in time, and thus corrupt the received data. This may be effectively handled by deploying Time Division Multiple Access (TDMA), that is to divide time frames into small segments and allocate each transmission a unique segment, in the Medium Access Control (MAC) layer [Cionca et al., 2008]. Furthermore, Forward Error Correction (FEC) methods can be deployed to improve link reliability in the Data Link Layer (DLL) [Jeong and Ee, 2003]. However, even if these techniques are used, unpredictable link failures due to changing radio conditions may still occur [Cerpa et al., 2005; Zhao and Govindan, 2003]. Hence, it is important to promote *robustness* against link failures of a network configuration. Again, in this case, the robustness is dependent on the paths taken as sensors sending data through multiple paths which share as few links as possible are likely to increase the chances of data being received at the base station.

Optimising energy efficiency in WSNs requires selecting the routes that use least energy links. This however may impose greater loads on some nodes that are part of the energy efficient routes and thus relay most data. From the perspective of optimising network lifetime, such loads should be distributed while mostly preferring energy efficient routes. In comparison, robust routes are those which share as few links as possible without any

regard to average or network lifetime. Hence, improving robustness is not directly related to the other objectives and may or may not be in conflict with them. Therefore, there are various trade-offs to consider while selecting routes with a view to optimising different objectives simultaneously.

Many routing approaches consider *single-path routing schemes*, where each node has a single path to the base station, due to their simplicity and efficient resource utilisation [Radi et al., 2012]. However, such approaches are somewhat limited in balancing the loads on nodes. This is because selecting one route per node and minimising the overlaps between different routes to reduce the loads on any particular node, would require selecting some longer energy inefficient routes. Furthermore, single-path schemes are unable to reap the benefits of using mesh networks fully as they are not capable of using alternative paths when node or link fails. In case of node or link failure, a single-path routing can be re-planned or re-optimised, and the network re-configured accordingly (see for example Rahat et al. [2014]). An alternative approach is *multi-path routing schemes*, where each node uses a number of paths to send data to base station, and thus it allows receipt of at least partial data when failure occurs. Practical resource limitations mean that only a small number of routes (two or three) may be allowed per node. In such a scheme, a proportion of messages is sent via each of the available routes; only one route is used for each individual message. Apart from being fault tolerant, this scheme is energy efficient as it allows better load balancing [Radi et al., 2012; Ganesan et al., 2001]. To use multi-paths, it is necessary to select the routes and the associated proportion of time each route should be used to send messages. As we show in this thesis, this time share problem may be modelled as a linear program for optimising robustness or network lifetime, and thus the optimal time shares for routes may be determined.

In this thesis, our goal is to provide general methods to locate a set of routes where each node has one or more routes to the base station that approximate optimal trade-offs between, firstly, average lifetime and network lifetime and, secondly, robustness and network lifetime. As we discuss in chapters 3 and 4, this combinatorial multi-objective problem is analytically intractable, and thus there are no known algorithms that will solve this problem in polynomial time. The population based approach of evolutionary algorithms, where many solutions can be obtained in a single run, works well for the fast approximation of the optimal trade-off in multi-objective problems. We therefore propose hybrid evolutionary methods which incorporate exact solutions to the time share linear program to estimate optimal trade-offs between objectives.

1.1 Novel Contributions

The major contributions of this thesis can be summarised as the following.

- We propose a hybrid evolutionary approach to estimate the optimal set of routes that trade-off between: energy efficiency and network lifetime, and robustness and network lifetime.

- We describe a novel robustness measure – the fragility – to quantify and compare the robustness of different solutions.

- We show how to improve the efficiency of the search and combat potential combinatorial explosion by using various novel search-space pruning methods which promote rapid optimisation. The methods proposed in this thesis are:

  k-shortest path pruning. Intuitively, shorter energy efficient paths are likely to generate overall good solutions. Therefore, we restrict the search space by choosing only the first k-shortest paths to the base station for each node.

  Max-Min Lifetime Pruning. When there are no limits on how many paths nodes may use to send data to the base station, a linear program (LP) may be solved to determine the optimal distribution of traffic in order to maximise the network lifetime [Chang and Tassiulas, 2004]. The resulting distribution can be used to reduce the graph by deleting unused edges; furthermore, applying k-shortest paths on this reduced graph enables the evolutionary search to select solutions with long network lifetimes.

  Braided and edge-disjoint path pruning. Braided and edge-disjoint paths are known to be highly resilient and energy efficient [Ganesan et al., 2001]. In addition to energy efficient k-shortest paths, we use braided and edge-disjoint paths to restrict the search space to regions likely to contain robust routes.

- We show that for a given set of routes an appropriate linear program may be solved to determine the optimal traffic distribution, i.e. the proportion of time each route should be used when there are multiple routes available, for network lifetime or robustness.

- The proposed methods are illustrated on a real network deployed in the Victoria & Albert Museum, London, UK. We show that we can successfully achieve a wide range of solutions displaying trade-offs between objectives. In particular, we achieve solutions with long network lifetimes close to the optimal lifetime that can be achieved if there are no constraints on the number of routes per node.

1.2 Thesis Structure

The rest of the thesis is structured as follows.

Chapter 2

We present a brief overview of wireless sensor networks, k-shortest path algorithms, and multi-objective optimisation problems. In particular, we discuss applications, challenges, and network topologies in wireless sensor networks. We also demonstrate k-shortest path algorithms in synthetic networks, and finally, we briefly present multi-objective optimisation problem formulation, solution techniques and performance metrics.

Chapter 3

In this chapter, we describe a model of wireless sensor networks, and derive a multi-objective problem relating to improving energy efficiency and network lifetime. We also show how this multi-objective problem can not be solved in polynomial time, and hence we describe a hybrid evolutionary method to solve it. We also show how the k-shortest path algorithm and a linear program for network lifetime (devised by Chang and Tassiulas [2004]) may be used to prune the search space while retaining good routes to combat combinatorial explosion. We extend the single-path approach to a constrained multi-path approach, and devise a linear program to find the optimal time shares between the constituent single-path schemes. Finally, we show that our hybrid approach can achieve a wide range of solutions including an estimation of the optimal trade-off between energy efficiency and network lifetime with one solution consisting of three paths per node achieving a lifetime within 1% of the optimal network lifetime.

The materials presented in this chapter have been published in [Rahat et al., 2014, 2015b].

Chapter 4

In this chapter, we turn our focus to solving another problem in wireless sensor networks: the problem of ensuring robustness in data delivery under uncertainty. We develop a fragility measure based on the maximum expected data loss in different routes to quantify robustness. We demonstrate the nature of this measure through a synthetic network. As network lifetime is often seen as more important than energy efficiency, we derive a multi-objective problem with a view to improve robustness and network lifetime simultaneously. As before, this presents an NP-hard problem, and again we devise a hybrid evolutionary method to solve it. In addition to using k-shortest path pruning and max-min lifetime pruning, we use braided and edge-disjoint path pruning to limit the search space. Here, we utilise unconstrained multi-paths, an extension to our previous approach, to approximate the optimal routings. We show that an appropriate linear program may be solved to determine the optimal time shares between the routes for robustness or network lifetime. We illustrate our method in both synthetic and real-world networks, and show that we can achieve a network lifetime within 1% of the optimal with a range of additional non-dominated solutions representing various levels of trade-off between robustness and network lifetime. We also describe potential avenues of improving local performance by using vertex enumeration within a deflationary process, and incorporating weighted sum method in the evolutionary process.

Some of the materials presented in this chapter have been submitted for publication in [Rahat et al., 2015a].

Chapter 5

This chapter summarises the hybrid evolutionary approach and the associated results presented in this thesis, and draws conclusions.

Chapter 2

Background

In this chapter, we present the background necessary to appreciate our proposed hybrid evolutionary approaches towards routing optimisation in wireless sensor networks. We present a generic overview of various concepts in this chapter, while the specific usages of different techniques and associated rationale are described in relevant chapters.

2.1 Wireless Sensor Networks (WSNs)

Research in WSNs started with the initiation of the Distributed Sensor Networks (DSN) program at the Defence Advanced Research Projects Agency (DARPA) around 1980 [Wang and Balasingham, 2010]. Since then, this field enjoyed a lot of attention from researchers and users, especially after recent advancements in processor and sensor technologies, which enabled the design of compact battery powered sensors, and thus made it a viable option for industrial remote monitoring applications.

2.1.1 Applications

Originally motivated from military applications, advances in low-cost sensor technology and communication networks have inspired many applications in different areas, which can be grouped in two major categories: monitoring and tracking [Chong and Kumar, 2003; Yick et al., 2008; Wang and Balasingham, 2010]. The generic monitoring applications are: environmental monitoring, location monitoring, process automation, etc.; while tracking applications mainly incorporate object, animal, vehicle or human tracking.

In contrast, Kulkarni et al. [2011] suggested a distinction among applications based on data management; they are:

Periodic Reporting

In periodic reporting, data is collected from sensor nodes at a predefined regular intervals, and thus it is particularly suitable for industrial monitoring, for example

in the pharmaceuticals, food or the heritage sectors, where it is essential to monitor environmental parameters continuously. In such a strategy, the expected data traffic and volume are known, which in turn helps in detecting faults and overall network management. Nonetheless, this requires constant energy drain in batteries due to regular transmissions.

Event Detection

In case of event detection applications, data is scrutinised locally at the sensor units and only reported when monitored parameters indicate the data is useful, i.e. an event of interest has occurred. This strategy is, for example, useful for security applications where a breach of perimeter is of concern. Infrequent transmissions also result in less energy consumption. However, the data traffic and volume can be sporadic and unpredictable.

Database-like Storage

Sensors in database-like storage networks store all data locally at the sensor units. The base station is responsible for querying and retrieving interesting data from a sensor. This strategy lends itself to long term data collection applications, where the data analysis is done off-line, perhaps to discover trends and patterns, for example in habitat monitoring of ducks [Madden et al., 2005]. In this case, sensors expend more energy in processing queries and storing data than in communication. Hence, the challenge is to efficiently store and process data.

In this thesis, we model WSNs and associated multi-objective problems based on static (fixed location) monitoring networks with sensors reporting data to the base station at fixed intervals, just as seen in museums where upto a few hundreds of sensors are usually deployed to monitor the surrounding environment of artefacts. Nonetheless, the techniques discussed here may easily be adapted for other applications.

2.1.2 Challenges in Sensor Network Design

Designing WSN protocols presents several difficult issues in the form of data processing, communication and sensor management [Chong and Kumar, 2003]. Essentially, this constitutes dealing with the ad hoc nature of WSN deployments, changes in sensor node locations, changes in topology or node connectivity, limited resources at nodes, etc. [Kulkarni et al., 2011; Gungor and Hancke, 2009]. The most significant challenges are posed by the limited resources at a sensor and the dynamic radio environment; a brief account of these is given below.

Resource Limitation

Wireless sensors are usually battery powered standalone devices with a limited energy source, computational processing power and storage facility [Gungor and Hancke, 2009]. Additionally, the replacement of batteries may not be straightforward, especially if the sensor is placed at a location that is difficult to access [Kulkarni et al., 2011]. Apart from measuring the environmental parameters, the major tasks

carried out by a sensor are: transmission and reception of data, storage of sensed data and routing information, micro-controller operation, and maintaining an on-board display. These tasks make demands on the energy and memory usages of sensors, and thus it is crucial to manage these resources through appropriate routing and communication protocols in order to ensure overall efficiency and longevity of the network.

Dynamic Radio Environment

WSNs are usually deployed in a dynamic radio environment – links between nodes may fail as the underlying radio environment changes due to physical infrastructure changes, passage of people, and radio interference from other sources or multi-path propagation of the same signal. These factors effectively corrupt data such that it is unrecoverable, especially for low power transmissions, and this generally prompts a repeat transmission from sensor nodes. As the change in these factors is often unpredictable, the link quality is expected to vary over time and space [Cerpa et al., 2005; Baccour et al., 2012].

To combat such adversities a range of techniques is used. For instance, Direct Sequence Spread Spectrum (DSSS) or Frequency Hopping Spread Spectrum (FHSS) techniques may be used to split the frequency spectrum and send data at different frequencies with a view to avoid collisions, especially in the presence of multiple radio sources on the same frequency band [Kohno et al., 1995; Chien et al., 2001; Akyildiz et al., 2002; Chehri et al., 2006]. Additionally, to access the communication medium with a view to avoiding congestion or collision, Time Division Multiple Access (TDMA) methods may be used for splitting transmission times and specifying unique time slot for each transmission [Cionca et al., 2008; Ye et al., 2004]. Alternatively, to efficiently recover corrupted data, encoding and decoding techniques, also known as Forward Error Correction (FEC) methods, such as convolutional coding or turbo coding, may be applied on data packets [Jeong and Ee, 2003; Schmidt et al., 2009; Berrou et al., 1993].

However, despite such efforts links may behave unpredictably [Cerpa et al., 2005; Zhao and Govindan, 2003]. Therefore, devising routes that each node should take to send data to the base station must be done considering the dynamic nature of the radio environment, and ensuring that data from the sensors reaches the base station under uncertainty, that is improving robustness, is of paramount importance.

2.1.3 Network Standards

Network standards describe the required topology and communication protocols for facilitating interoperability within sensor nodes. In case of WSNs, they are developed with a view to aid low power operation. Several standards are in place from different standardisation organisations. The Institute of Electrical and Electronics Engineers (IEEE), Internet Engineering Task Force (IETF), International Society for Automation (ISA) and HART communication foundation are among the leading regulatory organisations [Yick et al.,

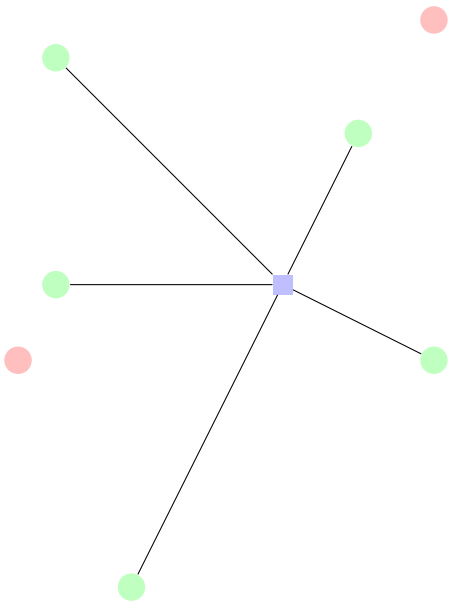


Figure 2.1 Illustration of a point-to-multi-point network topology. Sensor nodes (green circles) have direct links to the base station (blue square). Some nodes (red circles) may be outside the range of the base station, and thus they are disconnected from the network.

2008; Wang and Balasingham, 2010]. In this section, we discuss the common network topologies and relevant protocols used in practice.

Point-to-Multi-Point Network Topology

Point-to-multi-point topology is an ad hoc network topology that is common in WSNs. Generally, in these networks, all sensors operate in the same frequency band and the sensors send their data directly to a central node at a predefined rate. To facilitate the operation, this topology requires pre-configured routes, node specific IDs and security details [Young, 2008].

Often the fundamental protocol used is IEEE 802.15.4, which has been designed by the IEEE 802 organisation for Low-Rate Wireless Personal Area Networks (LRWPAN). It defines the physical layer and the data-link layer, and uses simple point-to-multi-point topology [Adams, 2006]. It emphasises low power, low complexity and low implementation costs, making it suitable for short range and energy efficient applications. Nonetheless, the detailed operation of this network topology largely depends on the manufacturers. In many instances, the manufacturers extend IEEE 802.15.4 to incorporate application specific ad hoc requirements.

In Figure 2.1, a simple point-to-multi-point network is shown. Here, nodes send only their own data, and thus it is a very low power solution. However, as nodes are required to be within direct range of the base station, often additional base stations or repeaters (nodes that relay received radio signals, often at a higher power than received signals) are required to establish full connectivity. Furthermore, if a link fails, no data can be retrieved from the affected node until the link is restored. To deal with such shortcomings, mesh network topologies have become popular in recent years.

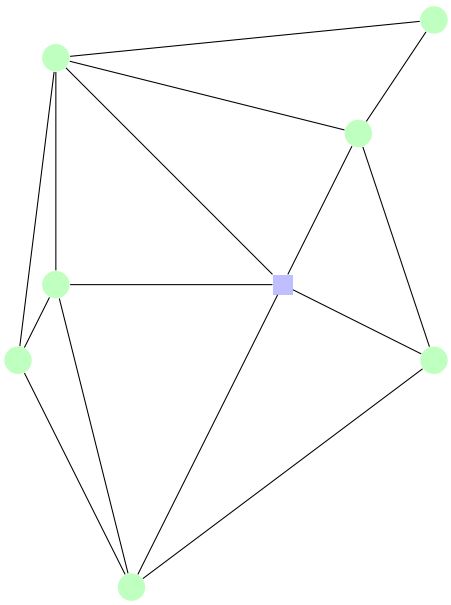


Figure 2.2 Illustration of a mesh network topology with all possible links. Sensor nodes (green circles) are linked to the neighbouring nodes or the base station (blue square). If a link to a neighbour fails, nodes may select another link to send data through a different neighbour provided that the neighbour has a route to the base station.

Mesh Network Topology

In a *mesh network topology* data is relayed from node to node en route to the base station. This topology extends the network range that can be covered by using multi-hops, that is multiple node-to-node relays. In addition, it provides an opportunity to use alternative routes in case links in existing routes fail, and thus provides an avenue to deal with a dynamic radio environment. However, this imposes a higher power and processing overhead at each node that relays other nodes' data, and therefore it is necessary to deploy an appropriate routing strategy to manage this additional demand and ensure energy efficiency.

Depending on where the routing decision is made, routing approaches in mesh networks can be divided into two groups: centralised and distributed [Kulkarni et al., 2011]. In centralised systems, routing information is calculated at the base station, and these routes are broadcast to the nodes for subsequent use. In contrast, in distributed networks nodes are responsible for calculating routes based on locally available information, e.g. link strengths to contiguous nodes. This additional responsibility imposes higher overhead from both energy consumption and memory (as it is necessary to save local connectivity information) perspectives in distributed systems. Also, as decisions are made locally, it may not be possible to achieve a globally optimal solution without sharing some information across the network (see for example Madan and Lall [2006]), and inevitably this imposes further transmission costs at nodes. Therefore centralised systems are often preferred for very low power sensor networks.

Common commercial high-level mesh network communication protocols are: ZigBee and WirelessHART [Yick et al., 2008; Wang and Balasingham, 2010]. Both these protocols use IEEE 802.15.4 as the basis for inter-node communication, and extend mesh network

concepts. ZigBee uses special repeater nodes (also known as routers) that not only retransmit all received signals, but also can work as a bridge between clusters of nodes and the base station. The use of such special nodes makes ZigBee a hybrid mesh network system. On the contrary, WirelessHART adopts a more generic mesh networking principle with only sensor nodes and the base station. It also deploys AES 128-block cipher for transmission security, and TDMA with time synchronisation to minimise collision and congestion. Clearly, this protocol has been designed with communication reliability in mind.

In this thesis, we consider a generic mesh network topology where nodes periodically (e.g. once every minute) send the measured data to the base station, potentially by relaying a message through one or more nodes via one or more routes. Such data reporting periods are repeated throughout the network lifetime: the time before any node exhausts its battery. This scenario is most common in industrial applications, especially for constant monitoring of locations [Kulkarni et al., 2011].

As we work with very low power sensor nodes, we use a centralised approach towards making routing decisions. At an initial mapping phase, the connectivity among nodes is discovered by *pinging* neighbouring nodes. This connectivity information is used to generate a complete network graph. The network graph is then used for routing optimisation purposes, which is the focus of this thesis. A solution in the context of routing optimisation is a set of routes where each node has one or more routes to the base station. As we optimise multiple objectives simultaneously, we end up with a range of solutions representing the trade-off between objectives; the novelty of this work is in devising general methods to approximate optimal trade-offs between objectives. It is then the network administrator's responsibility to select a suitable solution from the trade-off solutions. This solution is then broadcast to the nodes for subsequent use.

2.2 k-Shortest Path Algorithms

In this thesis we are interested in finding a set of routes where each node has one or more routes to send data to the base station with a view to estimate the optimal trade-offs between objectives. There may be many possible routes from a node to the base station. Hence, a set of routes is constructed by selecting one or more routes for each node from the collection of all possibilities, and thus each unique set of routes is a specific combination of routes. Therefore, finding sets of routes that optimally trade-off between objectives is considered as a combinatorial optimisation problem where different combinations have a particular quality on each objective.

In this combinatorial optimisation problem, the space to be searched for optimal sets of routes, that is the search space, grows rapidly. This is because the number of possible combinations of routes to choose from increases exponentially as the number of nodes grows. In other words, this problem is susceptible to combinatorial explosion. Therefore, to solve the optimisation problem quickly, we limit the search space by retaining sensible

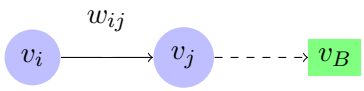


Figure 2.3 Route $S_i = \langle v_i, v_j, \ldots, v_B \rangle$ from v_i via v_j to the base station v_B. w_{ij} is the cost (e.g. distance) between v_i and v_j.

routes. Sensible routes in this context may incorporate routes that are short, energy efficient and reliable. This is based on the intuition that shorter paths are likely to be energy efficient and relatively reliable, because shorter paths require fewer transmissions. Hence data can be transmitted with lower overall energy expense, and fewer transmissions also result in higher cumulative probability of successful data transmission depending on the individual edge failure probabilities. A strategy for finding sensible routes is to focus on the k-shortest paths between each node and the base station. We discuss appropriate cost functions defining the length of each route in chapter 3. In this section, we discuss the k-shortest path problem and state-of-the-art algorithms to solve it.

The *shortest path problem* is a well known combinatorial optimisation problem. The goal of this problem is to find the route with the minimum distance between a source node and the base station with some predefined cost parameters (distances) at the edges. There are many effective algorithms for solving this problem, for instance, Dijkstra's algorithm [Dijkstra, 1959], Bellman-Ford algorithm [Bellman, 1958], A^* search algorithm [Hart et al., 1968], Floyd-Warshall algorithm [Floyd, 1962].

A natural extension of the shortest path problem is the k-shortest path problem, where the task is to find the k-shortest paths between a node and the base station, and rank them in increasing order [Eppstein, 1999; Hershberger et al., 2007]. This problem has warranted much research interest in the past few decades, since it was first scrutinised by Hoffman and Pavley [1959].

There are two variants of the k-shortest path problem. The distinction is characterised by the restriction of incorporating only simple paths, that is paths without loops. As it turns out, this problem with the simple path restriction is more challenging than with loopy paths. Nonetheless, several effective algorithms exist to calculate k-shortest simple paths. Some of the well-known algorithms are due to Yen [1971], Hoffman and Pavley [1959], Pollack [1961], Hershberger et al. [2007] and Brander and Sinclair [1995]. Yen's algorithm stands out as having the best worst-case upper bound computational time complexity for a directed graph. On the other hand, the problem is simpler to solve if simple paths are not required [Hershberger et al., 2007], and the algorithm with the best-known upper bound computational complexity to solve this problem is due to Eppstein [1999]. In this section, we describe both Yen's algorithm and Eppstein's algorithm, preceded by the fundamentals of graph representations.

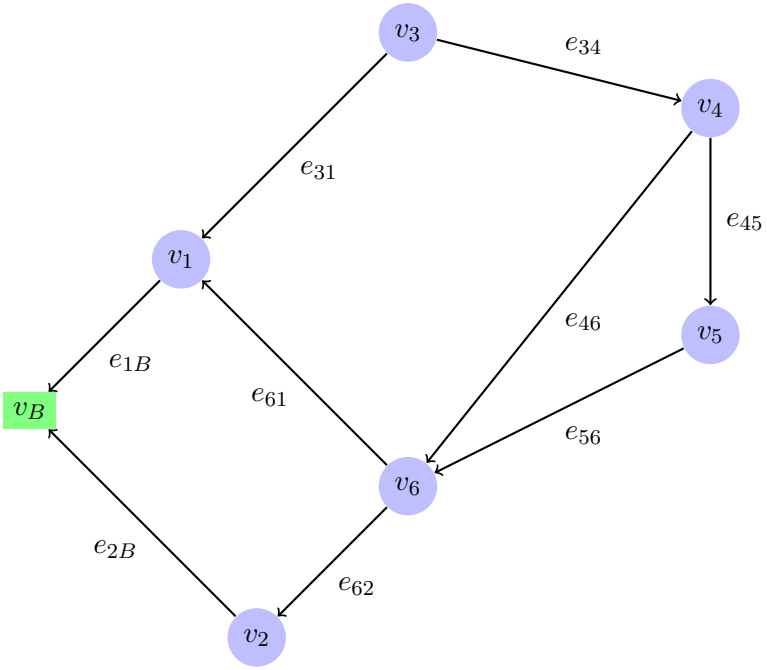


Figure 2.4 A directed network graph with six sensor nodes and the base station v_B. Nodes are represented by light blue circles, the base station is shown as a light green rectangle. Lines connecting the nodes are the edges, where an edge from node v_i to node v_j is labelled with e_{ij}. In this network, $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_B\}$ and $E = \{e_{1B}, e_{2B}, e_{31}, e_{34}, e_{45}, e_{46}, e_{56}, e_{61}, e_{62}\}$.

2.2.1 Graph Representations

Generally, a network consists of vertices and the arcs that connect them. The vertices are often referred to as nodes, while the connecting arcs are called edges or links. Therefore a network may be represented as the graph $G = \{V, E\}$, where V is the set of $N - 1$ sensor nodes v_i plus a base station v_B, and E is the set of M edges [Brander and Sinclair, 1995; Cormen et al., 2001].

An edge e_{ij} between nodes v_i and v_j is quantifiable in terms of an associated weight or cost w_{ij} (Figure 2.3). If $e_{ij} \Leftrightarrow e_{ji}$, then a graph is considered to be undirected, otherwise the graph is directed. The characteristics of the edge costs vary with the problem definitions. Costs can often be thought of a distance between nodes in which case the minimum cost route has a natural interpretation as the shortest path.

A route R_i is defined as a sequence of nodes that is followed to reach the destination node v_B, starting from a specific node v_i. For example, in Figure 2.4, one of the routes from node v_3 to node v_B is the sequence $R_3 = \langle v_3, v_4, v_6, v_1, v_B \rangle$. As routes involve edges, we can then calculate the total route cost by summing the edge costs:

$$H_i = \sum_{p=1}^{l-1} w_{R_{i[p]}, R_{i[p+1]}}. \tag{2.1}$$

Here, l is the length of the route and $R_{i[p]}$ is the p-th element of the route R_i. These route costs may then be used as a basis for comparison between different routes.

In a computational environment working with graphs requires special data structures.

Two particular structures are often used to define a network [Cormen et al., 2001]:

Adjacency list contains an array of nodes, where each member of the array is a list of neighbouring (connected) nodes. Therefore, the memory required to implement an adjacency list is $\Theta(M+N)$. In this structure, the information regarding connectivity between two nodes is not readily accessible. Figure 2.5 shows the adjacency list for the network presented in Figure 2.4.

List of Nodes	Adjacent to
v_1	v_B
v_2	v_B
v_3	v_1, v_4
v_4	v_5, v_6
v_5	v_6
v_6	v_1, v_2
v_B	-

Figure 2.5 Adjacency list for the network in Figure 2.4. In this list of lists each node of the network has a list of connected nodes to which it can send data.

Adjacency matrix is a square matrix of size $(N \times N)$, where the numeric indices of rows or columns are associated with nodes. Thus the i, j-th element of the adjacency matrix is the weight w_{ij} of the edge e_{ij}. A particular edge and the associated weights are easily accessible in this structure. A graph in adjacency matrix form can be described as follows [Cormen et al., 2001]:

$$G_{ij} = \begin{cases} w_{ij} & \text{if } e_{ij} \in E \\ \infty & \text{otherwise} \end{cases} \tag{2.2}$$

The memory requirements to implement an adjacency matrix is $\Theta(N^2)$.

Depending on the nature of the application, different data structures may be used. From memory and speed perspectives, the adjacency list is preferred when the application involve iterating through all edges. In contrast, the adjacency matrix is used when it is necessary to consider individual edges and their associated costs.

With this discussion of the basic notations and representations for computer networks, we present Yen's algorithm and Eppstein's algorithm next.

2.2.2 Yen's Algorithm

Yen's algorithm deals with the problem of finding k-Shortest Paths (KSP) on a graph with non-negative edge weights. It has the additional constraint that the paths are required to be simple (no loops are allowed) [Yen, 1971]. The algorithm is based on the idea that subsequent shortest paths share edges with previously found shorter paths. To exploit this idea, this algorithm maintains two lists: list A – the list of ranked shortest paths, and list B – the tentative list of candidate paths for the next shortest path. It starts with empty lists A and B. The shortest path between the source node and the base station

Algorithm 2.1 Yen's Algorithm for k-shortest paths:

Inputs

G: Adjacency matrix of the network
s: Source node index
d: Base station index

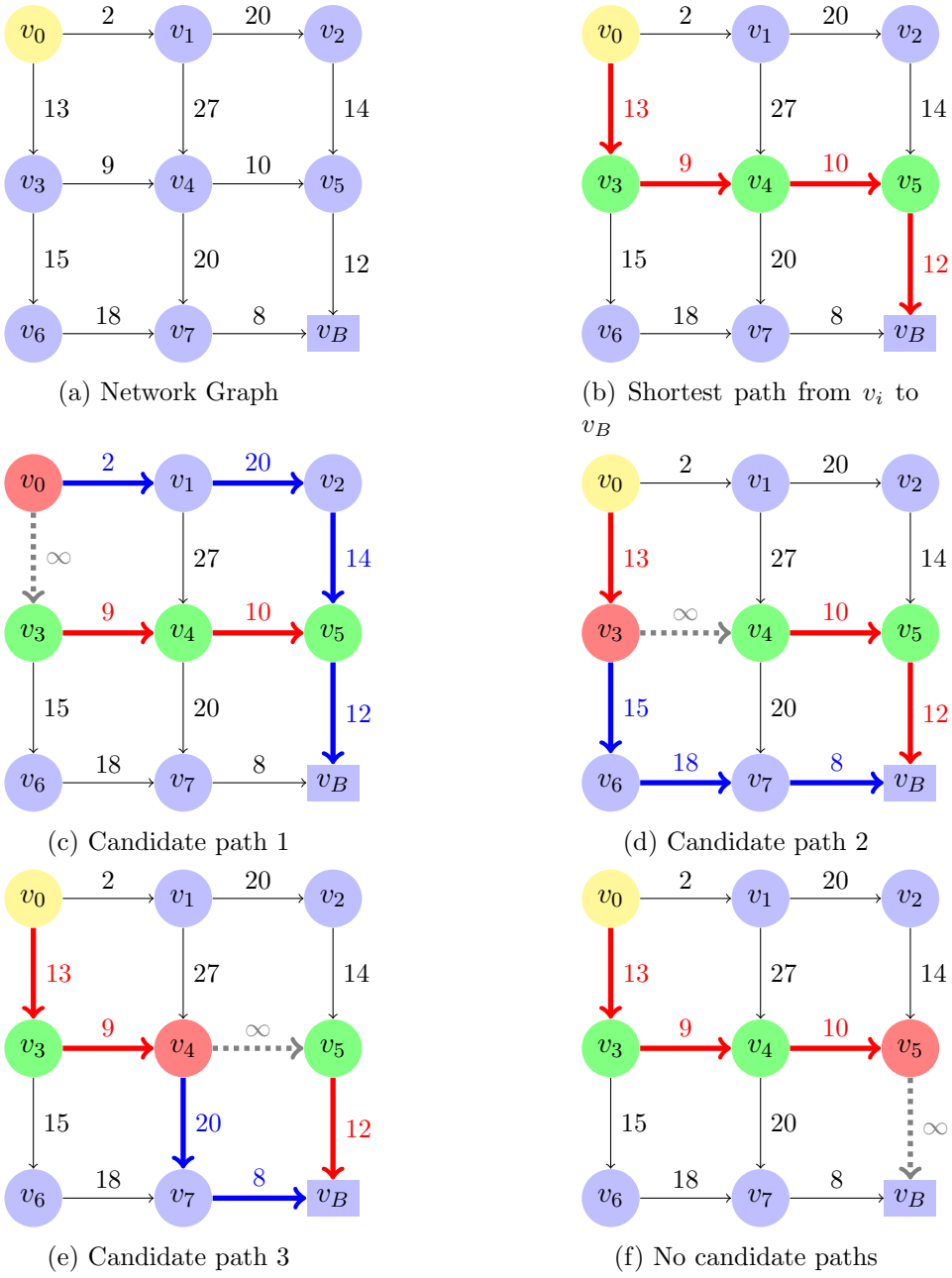
Steps

$A \leftarrow \langle\rangle$ ▷ Empty list of shortest path
$A_1 \leftarrow \langle s, \ldots, d\rangle$ ▷ Sequence of node indices in first shortest path [1]
$A \leftarrow A + A_1$ ▷ Append first shortest path to A
$G_l \leftarrow copy(G)$ ▷ Local copy of G
for $k = 2 \rightarrow K$ **do**
    for $i = 1 \rightarrow [len(A_{k-1}) - 1]$ **do**
        $n_c \leftarrow A_{k-1}[i]$ ▷ Current node
        $r_i \leftarrow A_{k-1}[0:i]$ ▷ Sub-path from source till current node in A_{k-1}
        for $j = 1 \rightarrow k - 1$ **do**
            $ind \leftarrow A_j.index(n_c)$ ▷ index of current node
            $r_j \leftarrow A_j[0:ind]$ ▷ Sub-path from source till current node in A_j
            if $r_i = r_j$ **then**
                $n_n \leftarrow A_j[i+1]$
                $G_{l(n_c,n_n)} \leftarrow \infty$ ▷ Remove edge
                $G_{l(:,n_c)} \leftarrow \infty$ ▷ Current node becomes unreachable
            end if
        end for
        $S_i \leftarrow \langle n_c, \ldots, d\rangle$ ▷ Shortest-path from current node till destination
        $B_i \leftarrow r_i + S_i$ ▷ New candidate for k-th shortest path
    end for
    $A \leftarrow A + \min(B)$ ▷ Append shortest-path among all paths in B
    $G_l \leftarrow copy(G)$ ▷ Restore original graph to local copy of G
end for

is then added to list A. The algorithm then loops through the following process until k shortest paths are found. For each path R_{id} in list A, for v_i, remove each edge e_{jk} in turn, where v_k and v_j are the head and tail of the edge. A root path to the tail is then $R_{id[:l]} = \langle v_i, \ldots, v_j\rangle$, where v_j is the l-th element in R_{id}. The shortest path R_j from v_j to the base station v_B is found with $e_{jk} = \infty$. The candidate path is then generated as $R_{i(d+1)'} = R_{id[:l]} + R_j$, and added to list B. In short, for each edge in each path in list A, a candidate path is generated and added to list B. Then the shortest path in list B is added to list A as the next shortest path. Algorithm 2.1 shows Yen's algorithm.

Figure 2.6 illustrates Yen's algorithm through an example of finding k-shortest paths in a synthetic network with $k = 2$. The network graph is given in Figure 2.6a. We want to find 2-shortest paths from v_0 to v_B. The first path is found by deploying Dijkstra's algorithm [Dijkstra, 1959], which yields the sequence, $R_{01} = \langle v_0, v_3, v_4, v_5, v_B\rangle$. As the shortest path in the network, this path is added to the list A. Now, for each edge in R_{01}, we remove it and generate a spur path from the tail of the edge. For instance, in Figure 2.6d, we remove the edge e_{34} with tail v_3 and root path $R_{01[:2]} = \langle v_0, v_3\rangle$. We then find the shortest path from v_3 to the base station v_B; this is the spur path $R_3 = \langle v_3, v_6, v_7, v_B\rangle$. The root

[1] Any suitable shortest path algorithm could be used to generate this sequence.



(a) Network Graph

(b) Shortest path from v_i to v_B

(c) Candidate path 1

(d) Candidate path 2

(e) Candidate path 3

(f) No candidate paths

Figure 2.6 Illustration of Yen's algorithm for finding 2-shortest paths from v_0 (solid yellow circle) to v_B (solid blue square) with a given graph in (a). The algorithm starts with the shortest path (red arrows) from v_0 to v_B in (b) (nodes in the path depicted with green); this path is added to the list of ordered shortest paths (list A). Edges from the shortest path are removed (grey arrows) in turn ((c)-(f)), and in each instance a candidate path is generated (blue arrows plus the path till solid red circles); each new path is added to the list of tentative paths (list B). Once all paths in list A have been considered, the shortest path in list B is added to list A as the next shortest path; in this case the next shortest path is (c).

path and the spur path are combined to generate a candidate path $R_{02'} = R_{01[:2]} + R_3 = \langle v_0, v_3, v_6, v_7, v_B \rangle$, and it is added to list B. Note that, in some cases a candidate path may not exist, see for example Figure 2.6f. Once all edges in R_{01} have been considered, the shortest path in list B, in this case the path in Figure 2.6c, is added to list A. Since, we have already found $k = 2$ paths, the algorithm returns list A, and terminates.

In this algorithm, the run time complexity has three components. The first component is the shortest path calculation, which is $\mathcal{O}((M + N) \log N)$ by using Dijkstra's algorithm

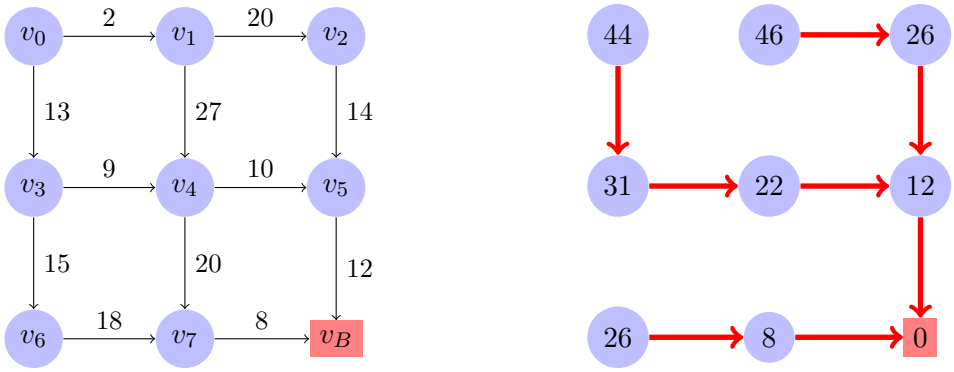


Figure 2.7 Illustration of shortest path tree. *Left*: a complete network graph G is given where the root node is the base station v_B (red). *Right*: the shortest path tree for G is depicted with red arrows, and the node specific shortest path cost is given inside a node.

with minimum priority queue [Cormen et al., 2001]. The second component is the length of the previous shortest paths, which at worst case can be N. The last component is k, as we have to iterate through previous shortest paths and find spur paths until we have k-shortest paths. Finding the next shortest path requires running shortest path algorithm once, and removing all edges in turn from the resulting path which may have N edges at worst. The combined run time for this procedure is $\mathcal{O}((M+N)N\log N)$. For finding k shortest paths, the process is repeated k times, and therefore the accumulated worst case running time becomes $\mathcal{O}((M+N)kN\log N)$, which is better than the reported original computational complexity of $\mathcal{O}(kN^3)$ [Yen, 1971; Brander and Sinclair, 1995].

This algorithm is designed to find the k-shortest simple paths between a source node and the base station. When the k-shortest paths for all nodes to the base station are required, this algorithm may be iterated for each node in turn. In the next section, we discuss Eppstein's algorithm, which deals with the k-shortest path problem without the limitation of simple paths. A straightforward modification to that algorithm may achieve simple paths as well.

2.2.3 Eppstein's Algorithm

Eppstein's algorithm has the lowest known run time complexity in calculating the k-shortest paths in a directed graph with non-negative edge weights and no restrictions on the paths being simple. The algorithm is based on three central concepts: shortest path tree, sidetrack costs and implicit representation of paths. We present a brief overview of these concepts first.

Shortest Path Tree

A tree in graph theory is defined as a subgraph of the complete graph where every pair of nodes has at least one route connecting them [Avis et al., 2005]. The shortest path tree problem extends this idea. In this problem the goal is to find a tree in the graph that is built with only the shortest paths from all nodes to a designated central or root node [Wu and Chao, 2004]. In the context of this thesis, the base station is considered to be the root node. A shortest path tree may be calculated

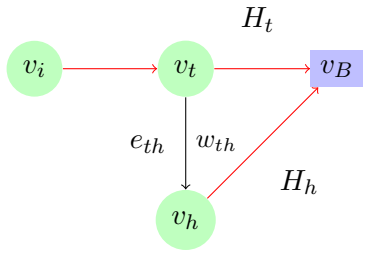


Figure 2.8 An edge e_{th} (black arrow) with edge cost w_{th} from node v_t to node v_h is not a part of the shortest path tree (red arrow). H_t and H_h are the shortest path costs for v_t and v_h respectively. The sidetrack cost at e_{th} is: $w_{th} + H_h - H_t$.

using Dijkstra's algorithm with Fibonacci heaps that has a run time complexity of $\mathcal{O}(M + N \log N)$ [Fredman and Tarjan, 1987; Wu and Chao, 2004]. An example of shortest path tree is given in Figure 2.7.

Sidetrack Costs

The sidetrack cost at an edge captures the cost saved by not being sidetracked from the shortest path tree. Let, for node v_t, the shortest path R_t on the shortest path tree have a total route cost H_t. Similarly, the shortest path R_h for node v_h has a cost H_h. Consider, an edge e_{th} from v_t to v_h which is not a part of the shortest path tree. Therefore, v_t may be sidetracked from the shortest path tree through the edge e_{th} taking the route R'_t instead of the shortest route R_t. By definition, the cost H'_t for new route R'_t would be greater than H_t as e_{th} is not a part of the shortest path tree. Thus the cost saved by v_t when it takes R_t without being sidetracked through e_{th} is given by:

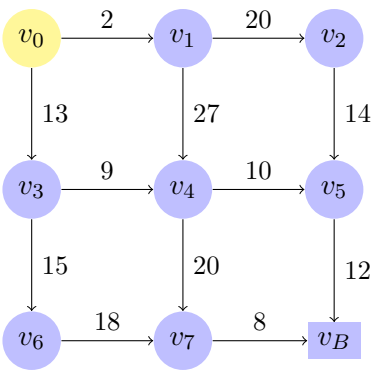
$$\Delta_{th} = H'_t - H_t = w_{th} + H_h - H_t, \tag{2.3}$$

where, w_{th} is the edge cost for e_{th} (see Figure 2.8). Clearly, the sidetrack cost at an edge on the shortest path tree would be zero as H_t and H'_t must be equal in this case.

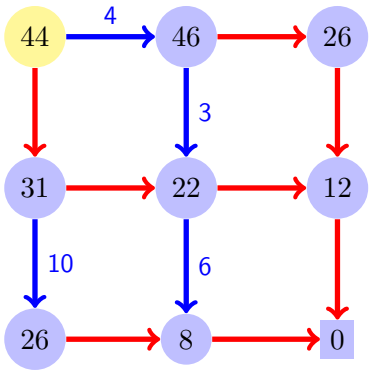
Note that by using the shortest path tree and the sidetracks costs, we can easily find new routes that are not in the shortest path tree for any node in the network, and the next shortest path must be among one of these new routes.

Implicit Path Representation

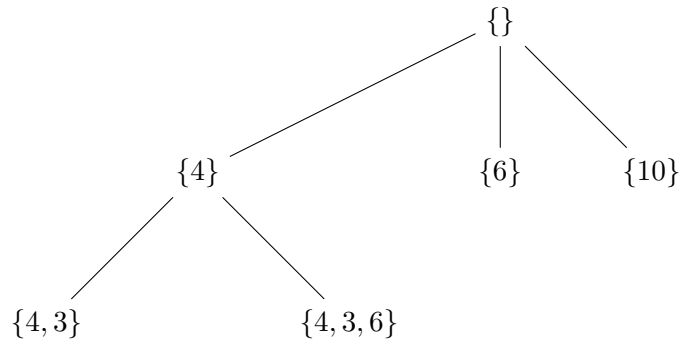
An explicit representation of paths in a graph is a sequence of nodes starting from the source node and ending at the base station. Eppstein [1999] argued that explicit representations of paths is computationally expensive, and hence he suggested an implicit representation of paths using sidetrack costs which is essentially a heap structure. The heap is formed with the edges that are not in the shortest path tree and ordered in terms of sidetrack costs. The root of this heap is the shortest path tree with null sidetrack cost. For building the rest of the heap, the rule is that every child in the heap is a possible sidetrack from the parent, and thus the child is essentially a particular sequence of edges where none of the edges appear in the shortest path tree (see Figure 2.9). Therefore, every pop operation, that is extracting the root of the heap, results in a particular sequence of edges. This sequence is essentially the least



(a) Network graph.



(b) Shortest paths tree (red arrows) with shortest route costs shown inside a node, and sidetrack costs (blue arrows).



(c) Heap built from sidetrack costs. Every pop operation extracts a sequence of sidetrack costs, and this sequence represents the next shortest path.

Figure 2.9 Building heap structure in Eppstein's algorithm for calculating k-shortest paths from v_i (solid yellow circle) to v_B (solid blue square) with given network graph in (a). The shortest path tree and the associated sidetrack costs are shown in (b), which is then used to build a heap structure in (c).

costly sequence of possible sidetracks within the current heap, and thus represents the next shortest path.

Eppstein's algorithm simultaneously generates the shortest path tree, the sidetrack costs and the implicit representation of the paths. Figure 2.9 demonstrates the sidetrack cost calculation and building the special heap structure. With the edge costs given in Figure 2.9a and the shortest path tree in Figure 2.9b, the sidetrack costs may be calculated. For instance, if v_0 sidetracks through edge e_{01}, the sidetrack cost would be: $w_{01} + H_1 - H_0 = 4$; all sidetrack costs can be calculated in this way. Starting from v_0, all possible sequences of these sidetrack costs may be pushed into a heap as in Figure 2.9c. Clearly, every instance the minimum is extracted from this heap, a particular combination of costs are produced, and thus using this sequence of costs a route may be constructed which would be the next shortest path. For example, extracting the minimum from the heap in Figure 2.9c, will produce the sidetrack cost $\{4\}$ which is associated with the sidetrack from v_0 to v_1 through e_{01}, so the next shortest route is: $\langle v_0, v_1, v_2, v_5, v_B \rangle$ with total cost of 48.

According to Eppstein [1999], the heap structure may be built with a worst case run time of $\mathcal{O}(M + N \log N)$. Then the k paths extraction, i.e. extracting the minimum from the heap

structure k times, has a run time complexity of $\mathcal{O}(k \log k)$. Therefore the combined computational complexity for this basic Eppstein's algorithm is of $\mathcal{O}(M + N \log N + k \log k)$. However, Eppstein explains a performance improvement technique using a tree decomposition method by Frederickson [1991], where the computational complexity becomes $\mathcal{O}(k + M + N \log N)$. Also, in recent times, practical performance improvements with the same upper bound complexity, have been described by Jiménez and Marzal [2003] and Aljazzar and Leue [2011].

Extracting Simple Paths

Eppstein's algorithm has no restrictions on the paths being simple. Nonetheless, it can be modified to produce simple paths. To achieve this, we ignore any loopy paths that are extracted from the heap and continue extracting the next shortest routes until we have k-shortest loopless paths or the heap is empty. This does not add any overhead on building the heap tree, as this is a common operation done as the first step of the algorithm.

In this section, we have reviewed the k-shortest paths problem and the state of the art solutions. Eppstein's algorithm, modified to discard loopy paths, is used to generate candidate routes in the evolutionary optimisation approaches described in chapters 3 and 4.

2.3 Multi-Objective Optimisation Problems (MOPs)

In this thesis, our aim is to find a set of routes with a view to improving two objectives simultaneously, that is to estimate the optimal trade-off between objectives. This gives rise to two-objective problems, which can be generalised as multi-objective problems. In this section, we review the basic concepts of multi-objective optimisation problems.

2.3.1 Optimisation and Multiple Objectives

Optimisation can be described as a problem of finding the best among a set of feasible solutions with respect to certain criteria. The feasibility is determined by the imposed constraints that must be satisfied for a solution to be considered as valid. If there is only one criterion to improve (either maximise or minimise), this is known as a *single objective* optimisation problem, and can be expressed as [Boyd and Vandenberghe, 2004]:

$$\min f_0(x), \tag{2.4a}$$

subject to:

$$f_i(x) \leq b_i, \quad i = 1, \ldots, k; \tag{2.4b}$$

$$f_j(x) = b_j, \quad j = k+1, \ldots, m; \tag{2.4c}$$

where the objective is to minimise $f_0(x) : \mathbb{R}^n \to \mathbb{R}$, and $f_i(x)$ is the i-th (equality or inequality) constraint of m constraints. A feasible solution vector $x \in \mathbb{R}^n$ resides in the set $\mathcal{X} = \{x \in \mathbb{R}^n : f_i(x) \leq b_i \wedge f_j(x) = b_j, i = 1, \ldots, k, j = k+1, \ldots, m\}$. An optimal solution to this problem is x^*, where $x^* \in \mathcal{X}$ and $f_0(x^*) \leq f_0(x); \forall x \in \mathcal{X}$.

There are various classes of optimisation problems [Boyd and Vandenberghe, 2004]. An optimisation problem is described as a *convex problem*, if the $f_0, \ldots, f_m$ functions satisfy the following conditions (in case of a minimisation problem):

$$f_i(\alpha x + \beta x') \leq \alpha f_i(x) + \beta f_i(x'), \quad \forall x, x' \in \mathcal{X}, \tag{2.5}$$

where $\alpha, \beta \in \mathbb{R}$. Note that the convexity property implies that global information may be derived from local information, that is if a local minimum is found, then it is also the global minimum [Boyd and Vandenberghe, 2004].

An important subclass of convex problems are *linear programming problems*, where all associated functions are linear, i.e. they satisfy the following conditions:

$$f_i(\alpha x + \beta x') = \alpha f_i(x) + \beta f_i(x'), \quad \forall x, x' \in \mathcal{X}. \tag{2.6}$$

If conditions (2.5) and (2.6) are not satisfied, then an optimisation problem is referred to as a *non-linear programming problem.*

In general, linear programs and convex problems can not be solved analytically. Nonetheless, there are effective algorithms to solve them. The most used approach to solve linear programming problems is the simplex algorithm due to Dantzig [1963]. Although this algorithm has an exponential worst case computational complexity, it performs well in practice and it has been shown to have polynomial smoothed complexity [Spielman and Teng, 2004]. Similarly, interior point methods may be used to solve convex problems, and the maximum number of operations required to achieve a specified accuracy is a polynomial of the problem dimensions [Karmarkar, 1984; Nesterov and Nemirovsky, 1994]. In contrast there are no known methods to solve non-linear programs in polynomial time [Boyd and Vandenberghe, 2004].

In many real world problems, it is not sufficient to optimise one objective; multiple objectives are required to be optimised simultaneously: this is known as a *multi-objective optimisation problem* (MOP). This can be formally expressed as [Zhou et al., 2011]:

$$\min \mathcal{F}(x) = (f_1(x), \ldots, f_m(x))^T, \tag{2.7}$$

where $x \in \mathcal{X} \in \mathbb{R}^n$ is a decision vector within the decision space $\mathcal{X}$, there are m objectives to minimise, and $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ with $\mathcal{Y}$ being the objective space (see Figure 2.10). Note that there may be equality or inequality constraints associated with the problem as well.

Here we are minimising all m objectives. In practice, some objectives may be required to be maximised while others are being minimised. Also, note that a maximisation problem can be expressed in the form of (2.7) without loss of generality [Deb, 2009].

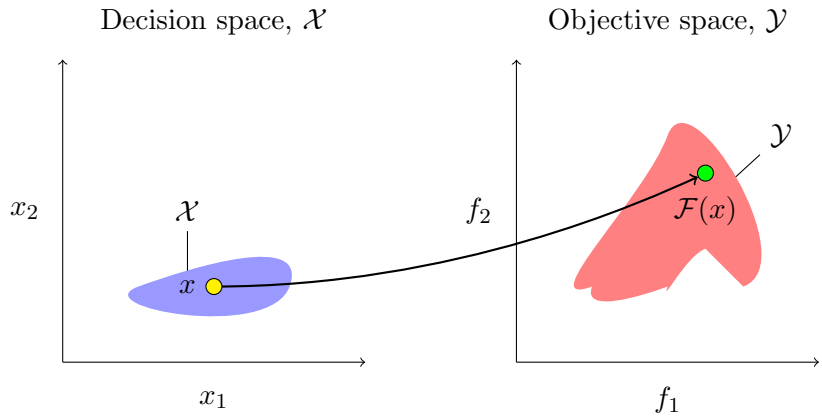


Figure 2.10 Objective space $\mathcal{Y}$ (red) is the image of the decision space $\mathcal{X}$ (blue). A solution $x = (x'_1, x'_2)^T$ (yellow dot) in decision space corresponds to a point $\mathcal{F}(x) = (f_1(x'_1, x'_2), f_2(x'_1, x'_2))^T$ (green dot) in the objective space. Solution quality is compared in the objective space.

2.3.2 Dominance and Pareto Optimality

In MOPs, the objectives are often conflicting: improving one objective can only be done at the expense of the other objectives [Coello Coello et al., 2007; Deb, 2009]. Therefore, a single solution that optimises all objectives simultaneously may not be available. In such cases only the best trade-off between the objectives exists, and these optimal trade-off solutions are known as the *Pareto optimal* solutions [Stadler, 1979]. A solution may then be chosen from the Pareto optimal solutions by the decision maker based on available extrinsic information, such as relative importance of the objectives.

The idea of Pareto optimality is based on the notion of dominance which is the standard basis of comparing two solutions in the objective space. A solution $x = (x_1, \ldots, x_n)^\top$ is said to *dominate* another solution $x' = (x'_1, \ldots, x'_n)^\top$ if x is wholly better than x'; this is written as $x \prec x'$. Formally, for the MOP in equation (2.7), $x \prec x'$ iff:

$$\mathcal{F}(x) \leq \mathcal{F}(x') \wedge \exists i \in \{1, \ldots, m\}, f_i(x) < f_i(x'), \tag{2.8}$$

where, $x, x' \in \mathcal{X}$, and $\mathcal{F}(x) \leq \mathcal{F}(x')$ represents element wise comparison [Coello Coello et al., 2007]. Figure 2.11 illustrates the dominance relationship.

There may be solutions $x' \in \mathcal{X}$, for which neither $x \prec x'$ nor $x' \prec x$, i.e. x is better in some objectives while x' is better in others. Then x and x' are incomparable and written as: $x \nprec x'$. These *mutually non-dominating* solutions are considered to be members of the Pareto optimal set if they are not dominated by any $x \in \mathcal{X}$. The Pareto optimal set is thus defined as:

$$PS = \{x \in \mathcal{X} | \nexists x' \in \mathcal{X}, x' \succ x\}. \tag{2.9}$$

The associated Pareto front in the objective space $\mathcal{Y}$ is described as:

$$PF = \{z = \mathcal{F}(x) | x \in PS\}. \tag{2.10}$$

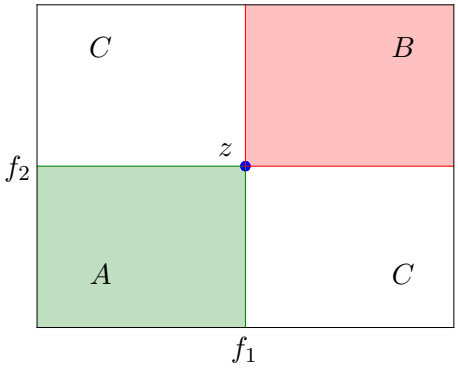


Figure 2.11 Dominance comparison in the objective space. Solution x has an objective value $z = \mathcal{F}(x)$ (blue dot). x dominates any solutions with objective values in area B (red), and x is dominated by any solutions with objective values in area A (green). The solutions with objective values in area C (white) are not directly comparable with x: these solutions are mutually non-dominated with respect to x.

2.3.3 Brief Overview of Solution Approaches

Traditionally, MOPs were solved using different scalarisation techniques [Ehrgott, 2006]. The most well-known among them is the weighted sum method in which the objectives are reduced to a single objective problem as follows:

$$\min \sum_{i=1}^{m} \lambda_i f_i(x), \tag{2.11}$$

where the weight λ_i represents the relative importance of the objective function $f_i(x)$. Solving this problem then results in a single solution, which relates to a particular trade-off on the Pareto front [Coello Coello et al., 2007]. The Pareto front may then be constructed by varying the λ_i, i.e. by varying relative importance of the objectives. This technique is however known to perform poorly for problems with non-convex Pareto fronts, and in fact evenly distributed weights fail to locate an even distribution of solutions across all parts of the Pareto front, even if the front is convex [Das and Dennis, 1997].

In the past few decades, *evolutionary algorithms* (EAs) have grown in popularity due to their ability to produce multiple solutions in a single run, and thus approximate the Pareto front rapidly. An approximation of the true Pareto front is often sufficient for real world problems [Coello Coello et al., 2007; Deb, 2009]. Rosenberg [1967] was the first to mention the use of a heuristic search approach towards solving MOPs. However, the first real application of EAs is due to Schaffer [1985]. Since then multi-objective optimisation using evolutionary algorithms (MOEAs) have received significant attention. There are many comprehensive survey papers. For example see Fonseca and Fleming [1995], Coello Coello [1999], Zitzler et al. [2004], Zhou et al. [2011], and Nedjah and Mourelle [2015].

There are many well known algorithm frameworks for MOEAs. Notable among these are Non-dominated Sorting Genetic Algorithm (NSGA) [Srinivas and Deb, 1994; Deb

et al., 2002], Pareto Archived Evolution Strategy (PAES) [Knowles and Corne, 2000], and Strength Pareto Evolutionary Algorithm (SPEA) [Zitzler and Thiele, 1999]. These frameworks mostly utilise a selection operator based on Pareto dominance and an iterative reproduction approach [Zhou et al., 2011].

In recent years, the focus of the research community shifted towards solving many-objective problems in which more than three objectives are optimised. Lücken et al. [2014] and Li et al. [2015] presented comprehensive surveys on recent evolutionary approaches to solve many-objective problems. Here we present a brief account of some important methods.

We note that in solving many-objective problems, the major difficulties are that the search ability deteriorates, and the number of solutions necessary to approximate the Pareto front increases exponentially [Ishibuchi et al., 2008]. With a view to combat such shortcomings, many methods have been proposed. An important class of techniques is decomposition based techniques in which the original problem is simplified into a collection of scalar functions [Lücken et al., 2014]. Hughes [2003, 2007] proposed a decomposition method called multiple single objective Pareto sampling (MSOPS, and MSOPS-II). In this method, each solution in the evolutionary population is ranked according to a set of target weights, and an individual's ranking is calculated using weighted max-min method. At the end of optimisation, the solution with most scores in the highest rank across all target weights is selected as the final solution. Multi-objective evolutionary algorithm based on decomposition (MOEA/D) is another popular decomposition method developed by Zhang and Li [2007]. In this technique, the scalarisation is achieved with a uniformly distributed weight vector. Each weight in the weight vector represents a particular scalar optimisation problem, and each individual in the evolutionary population is associated with improving one of these unique problems. To generate an appropriate weight vector, various methods, such as weighted sum, weighted Tchebycheff, boundary intersection and different combinations of them, have been used [Zhang and Li, 2007]. In addition to the decomposition methods there are indicator based approaches which use a quality indicator, such as the hypervolume indicator, as the fitness measure, and therefore the many-objective problem is converted into a single objective problem of extremising an appropriate quality indicator [Lücken et al., 2014]. In this context, Zitzler and Künzli [2004] have proposed an indicator based evolutionary algorithms (IBEA), which is a general framework. In this framework solutions are compared in pairs using a binary indicator, and only non-dominated solutions are evaluated. The hypervolume indicator may be used for indicator based approaches, for instance, S-metric selection-EMOA (SMS-EMOA) is a method that was designed to maximise the hypervolume indicator [Emmerich et al., 2005]. However, as calculating hypervolume in high dimensions is computationally expensive [Bringmann and Friedrich, 2008; Bradstreet, 2011], Bader and Zitzler [2011] described a fast hypervolume estimation method using a Monte Carlo algorithm, and they called the overall approach hypervolume estimation algorithm for multi-objective optimisation (HypE). Apart from that, Deb and Jain [2014] discussed a reference point based non-dominated sorting approach to deal with many objective problems, and this approach is known as NSGA-III.

In this thesis, in which two or three objectives are optimised, we use a straightforward elitist evolutionary algorithm combined with the deterministic search space pruning and

the solutions of linear programs; thus we develop a hybrid approach. We provide details of the algorithms in chapters 3 and 4.

2.3.4 Performance Metrics

To evaluate the performance of the hybrid evolutionary approach presented in this thesis, we use two methods:

Hypervolume Indicator [Zitzler, 1999]

The hypervolume indicator is the space covered between all function values $z = \mathcal{F}(x)$ and a predefined reference point r, where $x \in PS'$, an approximation of the optimal Pareto set. Usually, a reference point in the objective space is selected such that it is dominated by any function values from all approximation sets. Therefore, the indicator helps measure the coverage and the convergence quality of an approximation set. Thus it works as a basis for comparison between approximation sets.

Let, $a^i = \{y : y \in \mathcal{Y} \wedge y \succ z^i \wedge y \prec r\}$, then the hypervolume indicator with respect to a reference point r may be expressed as:

$$HV = vol\{\bigcup_i a^i | z^i = \mathcal{F}(x^i \in PS')\}. \tag{2.12}$$

In Figure 2.12, we give an illustration of the hypervolume indicator.

The advantage of using the hypervolume metric is that it can capture the dominance relationship between two approximation sets from the same simulation run precisely. This is because if an approximation set dominates another then it would have strictly better quality in terms of hypervolume in comparison to the latter [Zitzler et al., 2007]. The limitation of the hypervolume indicator is that the calculation of union of volumes takes exponential time in the number of objectives, and therefore there are no polynomial time algorithms to solve it [Bringmann and Friedrich, 2008; Bradstreet, 2011]. Although this may prove to be too expensive in very high dimensional problems, calculating the hypervolume in two or three dimensions is cheap.

Summary Attainment Surfaces [Fonseca and Fleming, 1996; Knowles, 2005]

With an approximation front, the objective space may be divided into two regions: firstly, the region where a point is either dominated by or equal to at least one member of the approximation front, and secondly, the region where no points are dominated by any members of the approximation front. Hence, there exists a boundary between these regions, and this boundary is known as the attainment surface [Fonseca and Fleming, 1996; Knowles, 2005].

The summary attainment surface (SAS) – also known as the empirical attainment surface (EAS) – is a statistical tool which shows the probabilistic distribution of the attainment surfaces achieved by a particular heuristic algorithm over many simulation runs. It is therefore used for estimating the performance of a heuristic algorithm, and it may be used to compare different algorithms.

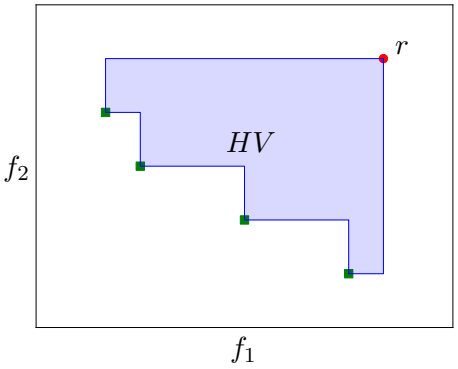


Figure 2.12 Hypervolume indicator (HV, blue area) in two dimensions. In this example, both objectives f_1 and f_2 are being minimised, and HV is the area dominated by the estimated Pareto front (green squares) with respect to the reference r (red dot).

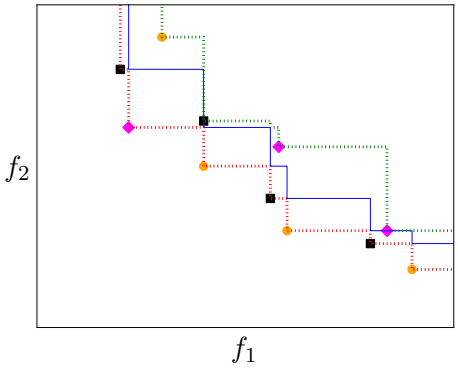


Figure 2.13 Schematic summary attainment surfaces for three simulation runs. Different Pareto front approximations are shown in black squares, magenta diamonds and orange dots. The red dotted line shows the 33.33% attainment surface, the blue solid line shows the *median* attainment surface, and the green dotted line shows the 100% attainment surface.

As stochastic optimisers only approximate the Pareto set, the solutions in final approximation may vary in different runs. The goal attainment of each run can be inspected using the attainment function, which is defined as the probability of an arbitrary point in the objective space $\mathcal{F}(x') = z' \in \mathcal{Y}$ being attained by solutions $x \in PS'$, where PS' is the final approximation Pareto set from the optimiser. In essence this is the probability that $\exists x \in PS' : x \preceq x'$.

SAS extends this idea to incorporate multiple runs [Fonseca et al., 2011]. Let an indicator function $I(\rho) : \mathbb{R}^n \to \{0, 1\}$, where $I(\rho) = 1$ if proposition ρ is true, otherwise $I(\rho) = 0$. Consider $PS' = \{x_i\}_{i=1}^k$ is a randomly drawn Pareto set approximation from some random non-dominated point set distribution, then the summary attainment surface is the discrete function $\alpha : \mathbb{R}^m \to [0, 1]$ with

$$\alpha(z') = \frac{1}{k} \sum_{i=1}^{k} I(x_i \preceq x'), \tag{2.13}$$

where $\mathcal{F}(x') = z'$ is an arbitrary point in the objective space.

Clearly, this is an estimate of the probability distribution of achieving certain quality in terms of the locations in the objective space. Thus this may be used to statistically evaluate the overall performance of a stochastic algorithm, and to compare different algorithms. An illustration of summary attainment surfaces is given in Figure 2.13.

2.4 Summary

In this chapter, we have reviewed some of the basic concepts regarding wireless sensor networks, k-shortest paths algorithms, and multi-objective optimisation problems that are

necessary to appreciate our proposed hybrid evolutionary routing optimisation method.

In this thesis, we experiment with WSNs in mesh network topology where each sensor is a node and sends data periodically to a base station. As we discussed, the major challenges in WSNs are the resource limitations at sensors and the unpredictability of radio environment. Due to the resource limitations, it is necessary to improve the energy efficiency in the network. In a mesh network topology, ensuring energy efficiency is equivalent to selecting routes with minimum energy links. This may impose undue burden on a few nodes that relay most traffic. Hence, it is necessary to distribute the loads in order to extend the time before any node exhausts its battery, that is the network lifetime. Additionally, WSNs are often deployed in unpredictable radio environment. It is therefore important to improve robustness of data delivery under uncertainty, and this objective has no direct relationship with the loads on nodes or energy efficiency. These unrelated and conflicting objectives may be considered as multi-objective problems, where good routes are sought to improve such goals.

The focus of this thesis is to locate a set of routes where each node has one or more routes to the base station with a view to optimising multiple objectives simultaneously. Here we consider two multi-objective problems: firstly, energy efficiency and network lifetime in chapter 3, and secondly, robustness and network lifetime in chapter 4.

Locating good routes is essentially a search problem, where the task is to find a set of routes from many possible combinations of routes. This is therefore susceptible to combinatorial explosion, that is as the number of nodes increases the number of possible combinations grows rapidly. To deal with this problem, we use various pruning methods, and one important pruning method uses k-shortest path algorithm, in particular modified Eppstein's algorithm discussed in this chapter, to reduce the search space by retaining energy efficient routes with an appropriate edge cost parameter. We discuss this method in more detail in chapter 3.

As discussed in this chapter, solving a multi-objective problem results in a range of solutions, and in this context a solution is a set of routes. Evolutionary algorithms are often used to solve such problems and to estimate the optimal trade-off between objectives. In the following chapters we show how hybrid evolutionary approaches may be used to solve the multi-objective routing optimisation problems in the reduced search space, and thus estimate the trade-off between different objectives.

Chapter 3

Energy Efficiency and Network Lifetime[1]

In this thesis, we consider a wireless sensor mesh network, where data is relayed from node to node *en route* to the base station. Mesh networks are preferred to *ad hoc* point-to-multi-point topologies as they are capable of extending the network range and provide alternatives when routes fail. However, mesh networks can be expensive in terms of energy consumption due to a higher overhead at each node for additional activities, namely relaying messages for other nodes and, in systems with distributed planning, calculating new routes. These additional activities can be severely detrimental to the overall life of the network, reducing the time before it requires servicing and battery replacement. It is desirable to minimise the energy consumption in the network, which is equivalent to maximising average lifetime [Rodoplu and Meng, 1999]. However, as Chang and Tassiulas [2004] indicated, using minimum energy routes throughout the lifetime of a network can be detrimental to the group of nodes that relays most routes. Hence, in addition to improving average lifetime, it will usually be important to maximise the time before the battery of the shortest-lived node is exhausted; this is the network lifetime. This routing optimisation problem is often referred to as the maximum lifetime routing problem [Chang and Tassiulas, 2004; Madan and Lall, 2006]. Devising routes in mesh networks therefore requires consideration of the trade-off between individual and system-wide battery lifetimes within the network.

Both efficient energy usage and maximum lifetime routing have attracted much research interest in recent years. These approaches can be divided into two groups: distributed and centralised [Kulkarni et al., 2011]. In distributed approaches, the responsibility for routing is distributed across the constituent nodes, i.e. the nodes are able to make decisions on routing locally. On the other hand, in centralised approaches, a centrally calculated route is broadcast to participating nodes.

Distributed approaches can provide good performance, and even optimal solutions. For instance, the distributed approach described by Rodoplu and Meng [1999] is guaranteed to

[1]The materials in this chapter have been published in [Rahat et al., 2014, 2015b].

converge to the minimum energy topology in a strongly connected network where the communication links are not vulnerable to environmental changes. In addition, Madan and Lall [2006] described a distributed approach that can locate the optimal routing scheme using a sub-gradient optimisation algorithm to solve the convex optimisation problem presented by maximum lifetime routing. Also, some heuristic approaches, such as reinforcement learning [Förster, 2007] and swarm intelligence [Bashyal and Venayagamoorthy, 2007], can be applied in a distributed fashion, although these only approximate the optimal solutions. An important consideration with distributed approaches is that nodes require sufficient computational power and storage to collect and store information regarding local connectivity and compute the best routes based on available information. In comparison, centralised approaches, which mostly incorporate variants of heuristic algorithms [Chang and Tassiulas, 2004; Xue et al., 2006; Islam and Hussain, 2006; Yetgin et al., 2012], require lower computational power and storage at the nodes, as most of the computation and storage is conducted by the central base station. Nonetheless, there is a system-wide overhead incurred in gathering connectivity information and broadcasting routing information. In this thesis we consider very low power nodes, each of which has limited computational power and storage. Routes are therefore computed at a mains-powered base station.

Most current evolutionary algorithm (EA) based energy-aware centralised systems consider energy expense. This is the case even for multi-objective routing optimisation, where energy expense is optimised with additional objectives describing different factors, such as quality of service, bandwidth, packet loss ratio, etc. [Xue et al., 2006; Yetgin et al., 2012]. However, as noted earlier optimising the overall energy expenditure of a network may be detrimental to the overall performance of the network, because often the goal is to prolong the network lifetime: the time before any battery needs replacing. Merely reducing the overall energy expenditure may place a large burden on a few nodes, resulting in the rapid exhaustion of their batteries. We therefore seek to optimise the network lifetime by modelling the charge held in their batteries and the energy expenditure at *each* node. Islam and Hussain [2006] and Kamath and Nasipuri [2011] have considered maximising only the network lifetime. Such approaches can improve the individual node specific energy state, but can be sub-optimal from system-wide perspective. We therefore seek to find the optimal trade-off between local and network-wide battery lifetimes.

In a mesh network where each node is able to route its messages to the base station via a number of different routes. Chang and Tassiulas [2004] have shown that a linear programming (LP) problem may be solved to obtain the proportions of messages that should be sent by each different route so as to achieve the maximum minimum lifetime. However, in very low powered battery networks the requirement to use many routes places an undue burden on the storage and computation that must be performed at each node. We therefore consider a routing scheme in which each node uses only a single path to the base station (this is known as *single-path* routing scheme). This routing strategy is widely used due to its simplicity and efficient resource utilisation [Xue et al., 2006; Jaffrès-Runser et al., 2010; Radi et al., 2012; Yetgin et al., 2012; Magaia et al., 2015]. In addition, we extend this concept to incorporate multiple single-path routing schemes (two or three) for the entire network to be changed at a few discrete times – this can be described as a

multi-path routing scheme, where nodes have multiple paths to send their data to the base station. Nonetheless, this is a somewhat limited version of multi-path strategy as nodes are not allowed to utilise their multi-paths independently as paths for all nodes change as a group. Ideally in a general multi-path scheme node specific route changes may be independent of other nodes – we discuss this in chapter 4.

The main contributions of this chapter are as follows.

- We present a hybrid evolutionary approach to approximate optimal routings with a view to maximise network lifetime and average lifetime; we also discuss why this problem is analytically intractable and the necessity for an evolutionary approach.
- To combat the potential combinatorial explosion presented by this multi-objective combinatorial routing optimisation problem, and promote rapid search, we use search space pruning using a k-shortest path algorithm and utilise the results of the linear program devised by Chang and Tassiulas [2004] to reduce the graph.
- The proposed method is shown to achieve close approximation to the path unlimited LP solution for a real network deployed in Victoria & Albert Museum, London.
- We also show that a different LP may be solved to find the optimum time share for which each routing scheme is active, given that entire network is allowed to change between single-path routing schemes.

In section 3.1 we describe the model of the wireless sensor network, derive the multi-objective problem, and discus why this problem can not be solved in polynomial time. In section 3.2 we present the maximum-minimum lifetime LP problem and discuss why maximising the average lifetime cannot be formulated as an LP problem. In section 3.3, we present search space pruning methods incorporating k-shortest paths and a graph reduction method using the LP solution to the maximum minimum lifetime problem. These are combined in section 3.4 where we present a hybrid evolutionary routing optimisation strategy. Based on this, in section 3.5, we demonstrate and discuss our findings in a real network deployed in the Victoria & Albert Museum, London. In section 3.6 we discuss how minimum lifetime of the system can be improved by using multiple routing schemes in an optimal time-shared manner. The chapter summary is presented in section 3.7.

3.1 Network Model and Multiple Objectives

In this section we model the WSN and derive a multi-objective problem in order to investigate how different routing schemes affect the trade-off between network and average lifetimes.

In the communication protocol used in this thesis, each node must send messages to the base station, perhaps by relaying a message through one or more other nodes. It is not acceptable for nodes to be disconnected from the network (i.e. to be unable to directly or indirectly route a message to the base station). In practical situations where this

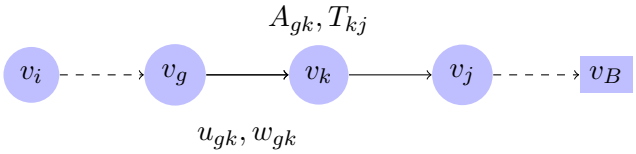


Figure 3.1 A route for node v_i: $R_i = \langle v_i, \ldots, v_g, v_k, v_j, \ldots, v_B \rangle$. The energy costs to send data from v_g to v_k are T_{gk} (transmission cost) at node v_g and A_{gk} (reception cost) at node v_k. The number of times edge e_{gk} is used by different routes (the edge utilisation) in the active routing scheme is u_{gk} with edge cost w_{gk} (the energy required to transmit data).

occurs, it is the role of the network engineer to distribute additional 'repeater' nodes to bridge communication gaps or to use nodes which can transmit at greater strengths. The connectivity information, that is which nodes can communicate with which other nodes, is gathered at an initial mapping phase; we call this a connectivity map. With this connectivity map the routing optimisation stage (the focus of this thesis) may be undertaken at the central base station with the assumption that the links are reliable. The result of this routing optimisation is a range of trade-off solutions, where each solution is a routing scheme that describes the communication activity between nodes during normal operation. The network engineer is then responsible for selecting a routing scheme based on extrinsic information; for instance the relative importance of network lifetime and average lifetime (relative importance may vary between deployments due to customer requirements or nature of the site). This selected routing scheme is then broadcast to all nodes, and it becomes the active routing scheme.

As discussed in section 2.2.1, a WSN is represented as a network graph, $G = \{V, E\}$, where V is the set of $N - 1$ sensor nodes v_i plus a base station node, v_B, and E is the set of M edges describing the connectivity map [Cormen et al., 2001].

In Figure 3.1, a route from node v_i to the base station v_B is described by the sequence, $R_i = \langle v_i, \ldots, v_g, v_k, v_j, \ldots, v_B \rangle$. We denote by $R_{i[p]}$ the pth element of the route R_i. Since we consider only single-path routes, a *routing scheme* $\mathcal{R}$ can then be described as a set of routes, one for each node in the network, to the base station:

$$\mathcal{R} = \{R_1, R_2, \ldots, R_N\}\,. \tag{3.1}$$

The energy required to transmit a message from v_i to the base station is the sum of the energies required to transmit a message between each of the nodes comprising the route:

$$H_i = \sum_{p=1}^{l-1} w_{R_{i[p]}, R_{i[p+1]}} \tag{3.2}$$

where l is the length of the route and w_{gk} is the energy required to transmit a message from v_g to v_k. Note that this generally involves energy expenditure at both the transmitting node and the receiving node, and will also involve expenditures for transmitting an acknowledgement. As noted above, we assume that the communication is reliable, but if an acknowledgement is not received from the receiver the message is resent; this additional

expense is not modelled, but if a link becomes unreliable, the routing may be re-optimised.

In many routing optimisation problems, such as shortest path problems, minimising a route's overall cost is desirable. The overall cost is found by summing the costs associated with each edge in the route. There are many well-known methods for minimising such costs, e.g. [Eppstein, 1999]. In this problem, however, we focus on the costs expended at the *nodes* themselves, rather than the edge costs. This is because it is energy expended at the nodes that depletes charge in the batteries and thus governs the lifetime of a node.

Let T_{kj} be the energy (charge) required at node v_k to send a message to v_j (Figure 3.1) and let A_{gk} be the energy required to receive a message from v_g at v_k. Then in one reporting cycle, the energy cost at v_k associated with relaying message in route R_i is:

$$C_i^k = A_{gk} + T_{kj}. \tag{3.3}$$

Note that, at the originating node v_i, there is no reception cost; hence, $C_i^i = T_{ir}$, where v_r is the immediately downstream node from v_i in R_i.

Also let the edge utilisation u_{gk} for a directed edge between v_g and v_k be the number of times in one reporting cycle that the particular edge is used by the routing scheme for a transmission between the associated nodes. Then considering all routes that use v_k to send messages in one reporting cycle, v_k receives messages from nodes with indices in the set $\mathcal{I}_k$ and sends data (including its own data) to nodes with indices O_k; the overall associated energy expense is

$$C^k = \sum_{g \in \mathcal{I}_k} u_{gk} A_{gk} + \sum_{j \in O_k} u_{kj} T_{kj}. \tag{3.4}$$

Clearly $\sum_k^n C^k = \sum_k^n H_k$ is equal to the energy cost across the whole network of sending a message from each node.

In order to calculate the lifetime remaining due to a routing scheme we require additional intrinsic information about the nodes, namely the charge q_k remaining in the battery and the quiescent energy consumption per reporting cycle B_k due to constant micro-controller operation, sensor measurements, running an on-board display, etc. The life of the current node therefore is modelled as

$$L_k = \frac{q_k}{(B_k + C^k)N_c} \tag{3.5}$$

where N_c is the number of reporting cycles per unit time (e.g. per year). We emphasise that $L_k \equiv L_k(\mathcal{R})$ is a function of all the routes which utilise v_k, and C^k is calculated from all routes in $\mathcal{R}$.

Our goal is to prolong the average life of the network, that is to minimise the total energy consumed, and to maximise the network lifetime, that is the time before any individual node requires its battery to be recharged or changed. We therefore arrive at the two

objective problem:

$$\text{Maximise} \quad f_1(\mathcal{R}) = \frac{1}{n}\sum_{k=1}^{n} L_k(\mathcal{R}), \tag{3.6}$$

$$\text{Maximise} \quad f_2(\mathcal{R}) = \min_{k\in[1,n]} L_k(\mathcal{R}). \tag{3.7}$$

In addition, it may be important to prolong the lifetime of one or more nodes v_k for $k \in \mathcal{U}$, because, for example, they are particularly inaccessible. In this case the two-objective problem is augmented with a third objective:

$$\text{Maximise} \quad f_3(\mathcal{R}) = \min_{k\in\mathcal{U}} L_k(\mathcal{R}). \tag{3.8}$$

The energy efficiency objective in (3.6) will ensure paths with the least energy consumption are selected irrespective of the load imposed on the nodes. As a consequence, certain nodes in the network may end up relaying most of the traffic (depending on the network structure). This is in conflict with the minimum lifetime objectives in (3.7) and (3.8), as to maximise these objectives it is better to distribute some load away from the nodes relaying most traffic. Additionally, (3.7) and (3.8) will be in conflict when the minimum lifetime node $v_m \notin \mathcal{U}$ and $\mathcal{U} \neq V$.

Solving this multi-objective problem may result in multiple solutions, as opposed to a single solution for single objective optimisation. In this case, there exists a set of solutions which are Pareto optimal; that is, there are no other feasible solutions available that improve performance on one objective, without a simultaneous decrease in at least in one other objective (see, for example, section 2.3.2 and Coello Coello et al. [2007]).

The dominance criterion is used to locate such solutions in the search space. The dominance criterion from a routing optimisation perspective is described as follows. In a multi-objective problem with m objectives, a routing scheme, $\mathcal{R}'$, is said to dominate another routing scheme, $\mathcal{R}$, denoted $\mathcal{R}' \succ \mathcal{R}$, iff

$$\begin{aligned} &f_i(\mathcal{R}') \geq f_i(\mathcal{R})\ \forall i = 1, 2, \ldots, m, \text{ and} \\ &f_i(\mathcal{R}') > f_i(\mathcal{R}) \text{ for some } i. \end{aligned} \tag{3.9}$$

Hence, we seek the maximal set of feasible routes which are mutually non-dominating, which is known as the Pareto set as discussed in section 2.3.2.

Locating Pareto set for this problem exactly would require searching all possible combinations of single-path routing scheme in the network. However, just counting all possible simple paths in a graph is known to be $\#P$-complete; thus the corresponding problem of listing all simple paths is NP-complete [Valiant, 1979a,b; Roberts and Kroese, 2007]. Hence, there are no suitable algorithm that can solve this problem in polynomial time and we therefore use hybrid evolutionary approach to approximate the optimal trade-off between average and network lifetimes.

3.2 Maximum Lifetime Routing and Energy Efficiency

Maximising the network lifetime – the time before at least one node exhausts its power source – is often referred to as the maximum lifetime routing problem [Chang and Tassiulas, 2004; Madan and Lall, 2006]. Chang and Tassiulas have identified the maximum lifetime routing as a linear programming problem. This is particularly useful as this LP problem can be solved in polynomial time. We use this solution in our hybrid evolutionary method, not only as a yardstick for our approach, but also as a means of directing the search. In this section, we present the LP formulation for maximum lifetime routing and discuss why the problem of maximising the average lifetime cannot be formulated as a linear program.

The LP problem formulated by Chang and Tassiulas [2004] incorporates multiple sets of source-destination pairs, and can be generalised when these pairs are active at different times with variable reporting rates in a multi-commodity setting. In this thesis, we consider a special case of their formulation, where all nodes send the same amount of data periodically at fixed intervals, e.g. a message every minute. This scenario is most common in industrial applications, especially for constant monitoring of locations. Also, we extend the approach with the inclusion of quiescent consumption at nodes as a practical consideration.

At the heart of the formulation of the LP problem lies the concept of network flow conservation: the outgoing flows from a node v_i must be equal to the sum of the incoming flows from other nodes and the flow generated at v_i. The network flow effectively indicates edge utilisations, i.e. how many times a particular edge has been used in a routing scheme. Hence, the flow conservation at v_i can be written in terms of edge utilisations.

$$\sum_{k \in \mathcal{I}_i} u_{ki} + U_i = \sum_{j \in O_i} u_{ij}, \tag{3.10}$$

where u_{ij} is the edge utilisation of the link from v_i to v_j and U_i is the flow generated at v_i, namely the data generated in each reporting cycle at v_i; the recipient node indices belong to set O_i and nodes transmitting to v_i have indices in the set $\mathcal{I}_i$.

A mathematical programming formulation for the maximum lifetime routing can be derived in the following way. The objective is:

$$\max_{\{u_{jk}\}} \left(\min_{i=[1,n]} L_i \right), \tag{3.11a}$$

which is the second objective of our multi-objective problem as presented in (3.7). The

associated constraints of this problem for $v_i \in V$ are:

$$u_{ij} \geq 0, \tag{3.11b}$$

$$N_c L_i B_i + \sum_{j \in O_i} N_c L_i u_{ij} T_{ij} + \sum_{k \in \mathcal{I}_i} N_c L_i u_{ki} A_{ki} \leq q_i, \tag{3.11c}$$

$$\sum_{k \in \mathcal{I}_i} u_{ki} + U_i = \sum_{j \in O_i} u_{ij}. \tag{3.11d}$$

Here the first constraint (3.11b) ensures that edge utilisations are non-negative, the second constraint (3.11c) expresses the fact that the energy usage of a node cannot exceed the remaining charge at the node, and the final constraint (3.11d) represents the flow conservation in the network.

The second constraint (3.11c) is derived from the definition of lifetime as provided in (3.4) and (3.5). Note that this expression is non-linear in u_{ij}, which means in general that the feasible region is non-convex and that methods guaranteed to find the optimum in polynomial time are not available [Boyd and Vandenberghe, 2004]; furthermore, the max-min objective function (3.11a) is non-smooth [Zang, 1980].

In the minimum lifetime case: $L_i \geq \min_{i=[1,n]}(L_i) = \mathcal{L}$, so L_i may be replaced with $\mathcal{L}$. Chang and Tassiulas [2004] therefore recast the problem in terms of $\hat{u}_{ij} = \mathcal{L} u_{ij}$ to obtain an LP problem that can be solved in polynomial time [Boyd and Vandenberghe, 2004]. The LP problem has the following objective:

$$\max_{\{\hat{u}_{jk}\}} (\mathcal{L}), \tag{3.12a}$$

subject to the following modified constraints:

$$\hat{u}_{ij} \geq 0, \tag{3.12b}$$

$$N_c \mathcal{L} B_i + \sum_{j \in O_i} N_c \hat{u}_{ij} T_{ij} + \sum_{k \in \mathcal{I}_i} N_c \hat{u}_{ki} A_{ki} \leq q_i, \tag{3.12c}$$

$$\sum_{k \in \mathcal{I}_i} \hat{u}_{ki} + \mathcal{L} U_i = \sum_{j \in O_i} \hat{u}_{ij}. \tag{3.12d}$$

This LP problem may be solved to obtain the minimum lifetime for the system, $\mathcal{L}$, and a set of compound edge utilisations $\hat{u}_{ij}$.

The energy efficiency of WSN systems is often viewed in terms of the average lifetime of the nodes. Maximising this ensures best possible usage of the energy available in the network as a whole [Rodoplu and Meng, 1999]. Similar to the minimum lifetime problem, the objective is a linear function of the edge utilisations (3.6) with the same constraints as for the minimum lifetime problem (equations (3.11b), (3.11c), and (3.11d)). Unfortunately, in this case the multiple quadratic constraints cannot be reformulated to obtain an LP problem meaning that efficient polynomial time methods cannot be employed. In addition, when the number of routes from a node to the base station is limited the Chang and Tassiulas [2004] method cannot be used to obtain an LP method for solving the minimum lifetime solution. We therefore turn to evolutionary methods to locate an approximate

Pareto front describing the trade-off between these objectives.

3.3 Search Space Pruning

The multi-objective optimisation problem described in section 3.1 is a combinatorial optimisation problem with, for practical WSNs, a vast number of potential solutions. The number of possible routing schemes, i.e. the search space size, depends on the number of available routes for each node in the system. For instance, in a network with n nodes excluding the base station let the number of available loopless paths from v_i to v_B be a_i. In this case, the number of possible routing schemes, i.e. the number of combinations of routes for individual nodes that can build the routing scheme, is:

$$Z = \prod_{i=1}^{N-1} a_i. \tag{3.13}$$

It is important for practical implementations that the optimisation process is fast. A way to improve the speed of optimisation is to sensibly prune the search space, while retaining important potential solutions. In this section, we describe two methods of pruning the search space: k-shortest path pruning and a max-min lifetime pruning, based on the maximum minimum lifetime solution, and discuss how they are used in approximating the Pareto set.

3.3.1 k-shortest Path Pruning

In order to combat the potential growth in the size of the search space as the number of nodes increases we limit the number of potential routes available to each node. More specifically, we limit the search to the space defined by the k-shortest paths for each node, where the metric defining the distance between nodes (the weight of the edges) is described below. This reflects our intuition that short paths to the base station are most likely to be energy efficient. We select from among several shortest path routes for each node because, if each node were to utilise its shortest path, nodes that occur in many of the 1-shortest paths would be disproportionately burdened.

Algorithms for discovering the shortest path between two nodes in a weighted graph are well known and the shortest path can be found in $O(M + N \log N)$ time [Yen, 1971; Eppstein, 1999; Brander and Sinclair, 1995]. However, as we noted above, the energy costs in this problem are associated with the nodes themselves rather than with the edges. We therefore weight the edges in the network graph to associate the energy cost at the nodes with the edges connecting them. Consider the nodes v_i and v_j. We define the weight of the edge between them as:

$$\omega_{ij} = \frac{T_{ij}}{q_i} + \frac{A_{ij}}{q_j}, \tag{3.14}$$

where, as above, T_{ij} is the energy required at v_i to transmit a message from v_i to v_j, A_{ij} is the energy required to receive a message at v_j from v_i, and q_i and q_j are the battery charges.

It is expected that if $q_i = q_j$, then $\omega_{ij} = \omega_{ji}$. This edge weighting models the fact that a high transmission cost can be borne by nodes with a high battery charge, but transmission is relatively expensive for nodes with low battery charge because each transmission will make a larger fractional depletion of the charge. Likewise, if a node is connected to mains power then transmissions are free, which is modelled by setting $q_i \to \infty$. We call the cost of a routing scheme calculated using the weights ω_{ij} the *composite cost*.

As we require diversity in the search space and the possibility of load balancing among nodes, we propose to evolve solutions from among the k-shortest paths for each node calculated with the composite cost (3.14). We presented a brief account of k-shortest path algorithms in section 2.2. In our implementation we have used Eppstein's algorithm modified to produce only simple or loopless paths as discussed in section 2.2.3.

We denote the m-th shortest route found for node v_i by R_i^m, for $m = 1, \ldots, k$. For a chosen k, we build a library $\mathcal{P}_K = \langle \{R_1^m\}_{m=1}^k, \{R_2^m\}_{m=1}^k, \ldots, \{R_n^m\}_{m=1}^k \rangle$ of paths, where each node has no more than k paths. The bound of k is not always attained because the number of loopless paths to the base station, v_B, for a node may be less than k, especially if it is close to v_B. A complete routing scheme R is then represented by a path for each node selected from one of the paths available in the library. Using this technique the reduced search space of possible solutions, Ω, is no larger than k^{N-1}.

Relationship between objectives and composite cost

In order to validate the use of the composite metric for choosing candidate routes, Figure 3.2 presents the correlation between the objectives and composite cost for 1000 randomly chosen routing schemes with random initial charge levels q_i at nodes for the real network deployed in Victoria & Albert Museum (see section 3.5). There is a fairly strong negative correlation between the average battery life $f_1(\mathcal{R})$ and the composite cost. This indicates that routing schemes selected with low composite cost via the k-shortest paths algorithm are likely to have long average lives. On the other hand, there exists a weak negative relationship between the minimum lifetime $f_2(\mathcal{R})$ and the composite cost. This is unsurprising since the minimum lifetime depends on a *single* node in the network and is highly dependent on the interaction between routes.

3.3.2 Max-Min Lifetime Pruning

The good correlation between composite edge cost and average battery lifetime, but poor correlation between composite edge cost and minimum battery life means that, although the k-shortest paths pruning retains good average lifetime routes, it may discard routes that would yield a long network lifetime. To obtain good candidate routes for long network lifetimes we solve the linear program in (3.12) to obtain the edge utilisations u_{ij} which maximise the minimum lifetime. We then consider the subgraph $G' \subset G$ obtained by deleting from G all the edges e_{ij} for which $u_{ij} = 0$. This subgraph G' is then used as the basis for generating a library $\mathcal{P}'_K$ of k-shortest paths with Eppstein's algorithm. The use

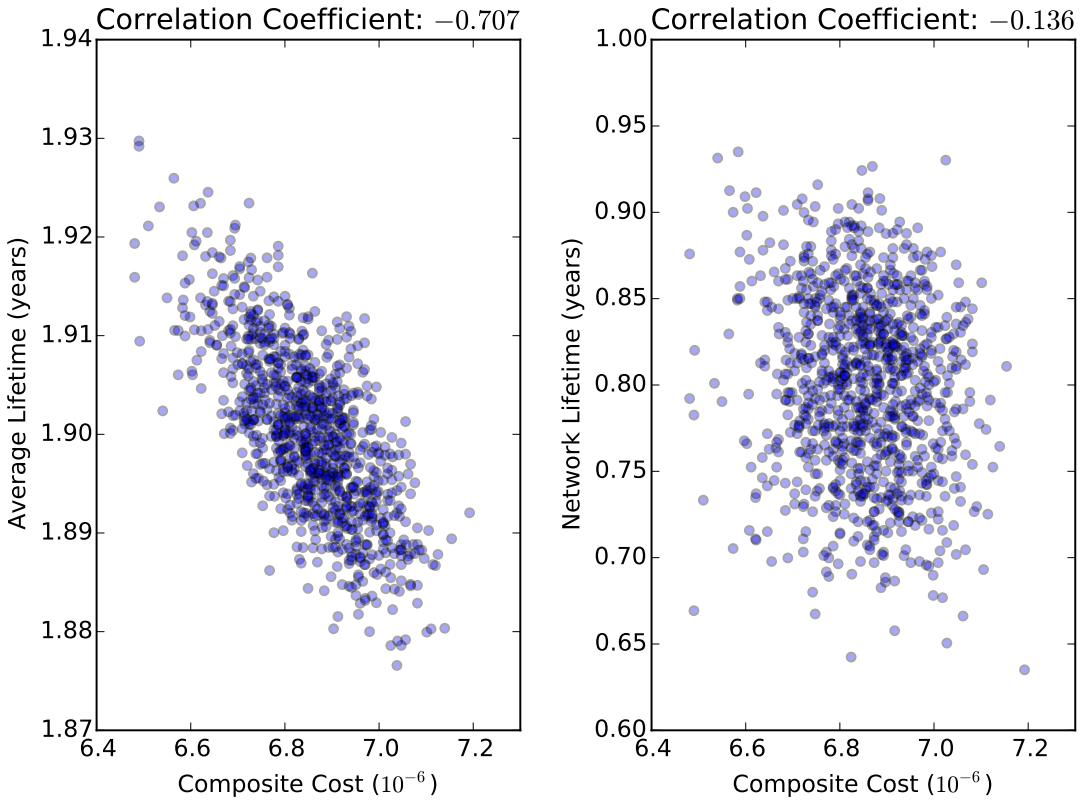


Figure 3.2 Correlation between objectives and composite cost. Scatter plots show the composite cost and average lifetime $f_1(\mathcal{R})$ *(left)* and minimum lifetime $f_2(\mathcal{R})$ *(right)* for 1000 randomly generated routing schemes with random initial charges at the nodes.

of G' prunes the size of the search space, but retains routes that are good for prolonging the minimum lifetime. We call this reduced search space Ω'.

The purpose of using such search space pruning is to retain routes from which to build good solutions. Here each method has its own specific goal: k-shortest path pruning for retaining solutions with better average lifetime and max-min lifetime pruning for retaining solutions with better minimum lifetime. We can then utilise the modified search space to rapidly obtain a sensible approximation of the optimal Pareto set. Although $G' \subset G$, the k-shortest paths derived from G' may be different from those derived from G. As a consequence, using the k-shortest path pruning may result in the search spaces Ω and Ω' being quite dissimilar if not completely disjoint.

3.4 Hybrid Evolutionary Approach to Routing Optimisation

The main focus of our approach is the rapid approximation of the trade-off front between minimum lifetime and average lifetime from infrequent maintenance and energy efficiency perspectives. In this section, we describe how we use hybrid evolutionary approaches in a multi-objective evolutionary algorithm to locate an approximately optimal set of routes.

Algorithm 3.1 Multi-objective routing optimisation of a battery powered mesh network.

Inputs

1: $\mathcal{P}$: Library of paths for each node
2: T : Number of iterations
3: s : Size of initial archive
4: μ : Perturbation rate
5: c : Crossover rate

Steps

1: $A \leftarrow \text{InitialiseArchive}(\mathcal{P}, s)$ ▷ Initialise random archive
2: **for** $i = 1 \rightarrow T$ **do**
3: $\quad \{\mathcal{R}^1, \mathcal{R}^2\} \leftarrow \text{Select}(A)$ ▷ Select two parent solutions
4: $\quad \mathcal{R}' \leftarrow \text{Crossover}(\mathcal{R}^1, \mathcal{R}^2, c)$
5: $\quad \mathcal{R}'' \leftarrow \text{Perturb}(\mathcal{R}', \mu, \mathcal{P})$ ▷ Mutation
6: $\quad A \leftarrow \text{NonDominated}(A \cup \mathcal{R}'')$ ▷ Update archive
7: **end for**
8: **return** A ▷ Approximation of the Pareto set

As described in the previous section, we prune the search space in two ways to obtain path libraries, $\mathcal{P}_K$ and $\mathcal{P}'_K$, of potential routes for each node in the network. We represent a solution as a vector of indices into these libraries, one index for each node, so that a complete solution $\mathcal{R}$ describes a route for each node to the base station. One optimisation strategy would be to select a route for each node from $\mathcal{P} = \mathcal{P}_K \oplus \mathcal{P}'_K$. However, as we demonstrate below, it is more effective to optimise in two stages. In the first stage, we perform two separate optimisations, one selecting routes from $\mathcal{P}_K$ and the other selecting routes from $\mathcal{P}'_K$. In the second phase routes are selected from $\mathcal{P} = \mathcal{P}_K \oplus \mathcal{P}'_K$. This phase is initialised from the non-dominated archives resulting from the two first stage optimisations.

Optimisations for both stages use a genetic algorithm, with an unconstrained Pareto archive to reap the benefits of better convergence properties [Fieldsend et al., 2003]. Algorithm 3.1 describes this multi-objective optimisation process in more detail.

Solutions, $\mathcal{R}$, are represented by vectors of n integers; the ith element of the solution indexes one of the k-shortest paths found for node v_i. In the initialisation step, we generate at random a population from the pruned search space, Ω. (The search space is Ω' when using the max-min pruning, but we describe the algorithm in terms of Ω.) This population comprises randomly-chosen routing schemes where the member routes are chosen from the k-shortest paths for each node; thus for node v_i the shortest paths are selected from $\{R_i^m\}_{m=1}^k$. Also, we include in the initial population the routing scheme which uses the first shortest route for each node. The initial archive of non-dominated solutions A is the maximal non-dominated subset of this random population. At any step of the evolution, A is the current approximation of the Pareto set within the particular chosen search space.

During the evolution process, we randomly select two routing schemes. These parent routing schemes are then crossed-over to yield a single offspring $\mathcal{R}'$: the route for each node v_i is selected either the first or second parent with probability c and $1-c$ respectively, independently of the other nodes. This child is then perturbed by choosing new routes

(uniformly at random from $\mathcal{P}_K$) for a proportion μ of the nodes in the network. The perturbed child, $\mathcal{R}''$, is compared against the members in the archive: if it is not dominated by any of them then it enters the archive and any elements of A which are dominated by the new solution are removed from A. In this fashion only the non-dominated routing schemes are preserved in the archive and the archive can only approach the true Pareto set. The process of evolution continues for a fixed number of episodes[2]. Alternatively, another termination criterion, such as a specified minimum dominated hypervolume [Zitzler, 1999], may be employed.

We denote the estimated Pareto sets resulting from the separate optimisations using $\mathcal{P}_K$ and $\mathcal{P}'_K$ as A and A' respectively. As illustrated below, the archive A resulting from optimisation using $\mathcal{P}_K$ tends to favour solutions with long average lifetimes, whereas A' resulting from optimisation using $\mathcal{P}'_K$ tends to yield solutions with long minimum lifetimes. However, solutions with intermediate average and minimum lifetimes may be scarce in $A \cup A'$. The second phase of the optimisation remedies this by employing the same optimisation algorithm, but now selecting paths from $\mathcal{P} = \mathcal{P}_K \oplus \mathcal{P}'_K$. This optimisation is initialised from the maximal non-dominated subset of $A \cup A'$; that is $A''_{init} = \text{NonDominated}(A \cup A')$, where NonDominated$(\cdot)$ is the function that returns the maximal subset of non-dominated elements of X:

$$\text{NonDominated}(X) = \{\mathcal{R} \in X \mid \nexists(\mathcal{R}' \in X \wedge \mathcal{R}' \prec \mathcal{R})\}. \tag{3.15}$$

We denote the archive resulting from the second optimisation by A''.

The overall hybrid evolutionary approach can be summarised as follows:

1. Gather the connectivity map, and construct the network graph G.
2. Solve the LP problem for maximum lifetime routing (3.12) to find the optimal network lifetime $\mathcal{L}$ and edge utilisations u_{ij}.
3. Search space pruning.
   a) Use u_{ij} to generate the reduced graph G' by selecting edges where $u_{ij} > 0$.
   b) Apply k-shortest path pruning to G and G' creating path libraries $\mathcal{P}_K$ and $\mathcal{P}'_K$ respectively.
4. First stage of routing optimisation. Apply Algorithm 3.1 using $\mathcal{P}_K$ and $\mathcal{P}'_K$ separately. This results in archives A and A', which are the approximations of the Pareto set in the relevant search space.
5. Initialise the second stage optimisation with $A''_{init} = \text{NonDominated}(A \cup A')$.
6. Second stage of routing optimisation: Apply Algorithm 3.1 using $\mathcal{P} = \mathcal{P}_K \oplus \mathcal{P}'_K$. On termination, we have the estimated Pareto set A''.

[2] As exactly one child is produced per episode, the number of function evaluations is equal to the number of episodes.

Once the evolution process is finished, the decision maker may manually choose the operating point using the final approximation of the Pareto front A''. The chosen routing scheme is sent to the nodes via the base station and becomes the active routing scheme.

3.5 Illustration

A real network deployed in the Victoria & Albert Museum, London, is used to illustrate the proposed approach.[3] In a controlled environment over a vast area, such as Victoria & Albert Museum, it is essential to monitor temperature and humidity in galleries and display cases for the preservation of the artefacts. Compared to wired networks, battery powered WSNs carry huge advantages in deployment cost and flexibility.

The network incorporated 30 sensor nodes and a base station, spanning five floors within an approximate area of 35,000 m^2. The thick, solid walls provide a challenging radio environment whose characteristics vary with the passage of visitors through the galleries.

3.5.1 Baseline Optimisation

The connectivity map G was built in an initial mapping phase in which nodes *pinged* each other using a range of baud rates and powers to discover the minimum energy configuration for communication between those nodes within radio range of each other.

Solving the maximum minimum lifetime LP problem results in a network lifetime of 1.38 years, together with a set of edge utilisations. These edge utilisations were used to calculate the average lifetime of the system, which is 1.69 years. They also enable the extraction of the sub-graph G'. We emphasise again that this network lifetime is only achievable through the use of multiple routes from each node to the base station, whereas we are constrained to seek a routing in which each node uses only a single route to communicate with the base station. Clearly, the requirement to use a single route means that battery loadings cannot be shared optimally so that a single routing scheme will have a worse (shorter) maximum lifetime than that given by linear programming.

Applying subspace pruning using $k = 10$ shortest paths results in libraries $\mathcal{P}_K$ and $\mathcal{P}'_K$ from each of which 10^{30} solutions could be constructed. In each search space, the initial population was built with 100 randomly chosen solutions from the relevant subspace together with the solution consisting of the shortest route for each node. The initial archive was then found by extracting the maximal set of non-dominated solutions in this population.

In the evolutionary optimisation we used $\mu = c = 0.1$ as the perturbation and crossover rates; these rates were chosen after a short empirical study on simulated networks, but the performance of the optimiser is relatively insensitive to their precise values. Using

[3] Data on the network to allow comparison with this work can be found at `http://emps.exeter.ac.uk/computer-science/wsn/`

the dominated hypervolume measure [Zitzler, 1999], the Pareto sets for the first stage optimisation were well converged after 150000 function evaluations (Figure 3.6). The second stage optimisation used the same μ and c as the first stage. This evolution was run for 500000 function evaluations, resulting in the final approximation of the Pareto set A''.

A single core Python implementation takes about 2 minutes to complete each of the initial 150000 iterations on a 2.5 GHz machine. The 500000 iterations in the second stage takes about 4 minutes to complete. Consequently, the routes for new installations can be found readily, making this approach particularly feasible as part of a real system.

Figure 3.3 shows the estimated Pareto fronts generated during the initial stage of optimisation. As the figure suggests, in both search spaces, Ω and Ω', there is a wide range of routes which trade-off the average lifetime of the network against the time before any single node exhausts its battery. We note that the optimisation has resulted in routing schemes that are substantially better, in terms of both average and network lifetime, than routing schemes utilising randomly chosen routes from the 10 shortest paths in both Ω and Ω'. Indeed, for both search spaces, the routing schemes with the longest network lifetime have a better average lifetime than any of the random solutions for that search space. Interestingly, the shortest paths only solutions $\mathcal{R}_c$ and $\mathcal{R}'_c$ are very close to the estimated Pareto front in both spaces. We include $\mathcal{R}_c$ and $\mathcal{R}'_c$ in the initial archives for the relevant search spaces.

We note that for the Victoria & Albert Museum network 13 of the 30 nodes have a possible path which appears in both $\mathcal{P}_K$ and $\mathcal{P}'_K$, although no node has more than 3 paths common to both libraries. Since 17 of the nodes therefore have no paths appearing in both libraries, it is not possible to construct a complete routing scheme, $\mathcal{R}$, that exists in both libraries and the search spaces Ω and Ω' are completely disjoint. Also the fronts occupy different parts of the objective space. Many of the solutions in A' (resulting from the max-min pruning) are dominated by solutions in A, which resulted from k-shortest path pruning of the full graph G. Nonetheless, the utility of using the min-max pruning is shown by the 11 solutions that achieve longer network lifetimes than any solution found in Ω.

Combining A and A' leaves a large gap in the estimated Pareto front for intermediate average lifetimes. To remedy this, the second stage optimisation is initialised from NonDominated($A \cup A'$) and the union of the two pruned search spaces is available for constructing new solutions. The result of this second optimisation is shown in Figure 3.4.

As can be seen from Figure 3.4, A'' is very close to A in the high average lifetime region, but combining paths from $\mathcal{P}_K$ and $\mathcal{P}'_K$ has resulted in solutions being found with good network lifetimes and better average lifetimes than those in A'. The linear programming solution (that solves for many routes per node rather than a single route) is marked by a grey triangle in the figure and this upper bound on the minimum lifetime, $\mathcal{L} = 1.38$ years, is indicated by the horizontal line. The best minimum lifetime solution when nodes are constrained to use a single route found by the evolutionary optimiser has a minimum lifetime of 1.29 years. This solution improves the shortest-path only routing scheme by

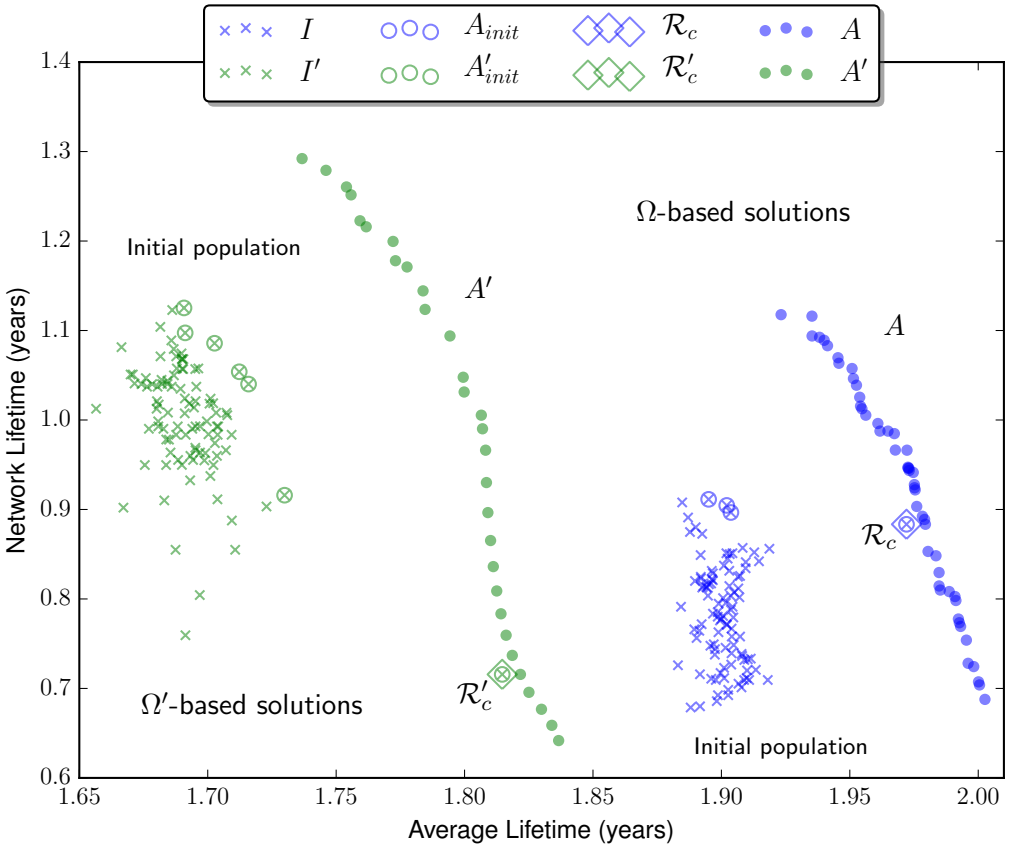


Figure 3.3 First stage optimisations using two pruned search spaces Ω (blue) and Ω' (green) for the Victoria & Albert museum network. Initial randomly generated solutions and the optimised fronts are shown. Green coloured symbols represent solutions for Ω', while blue indicates solutions for Ω. Initial populations (I in Ω and I' in Ω') are shown with crosses; ringed crossed show the non-dominated initial archives, A_{init} and A'_{init}. The shortest paths only solutions ($\mathcal{R}_c$ in Ω, and $\mathcal{R}'_c$ in Ω') are indicated with diamonds.

approximately 5 months in minimum lifetime at an expense of 2 months in average lifetime. In contrast, the solution with best average lifetime improves the shortest-path only solution by 7 days in average lifetime, while the minimum lifetime is worse by 2 months, which is a rather small gain at a large expense.

Figure 3.5 shows a comparison between the two-stage approach and a single-stage approach in which solutions may be constructed from $\mathcal{P}_K \oplus \mathcal{P}'_K$ throughout the entire optimisation. As can be seen from Figure 3.5, the estimated Pareto fronts resulting from the two strategies mostly overlap, except at the extreme maximum network lifetime/ minimum average lifetime end. Here, the strategy of evolving solutions using a path library that favours good network lifetime solutions has paid off, yielding additional solutions with long network lifetimes. To generate this figure, the single stage strategy was allowed to run for as many objective function evaluations as the two stage strategy. This advantage is also apparent from the hypervolume progressions of the single-stage and the two-stage approaches as depicted in Figure 3.6. Despite the additional complexity of the optimisation algorithm, we therefore adopt the two-stage approach.

We also investigate the utility of using the search space pruning methods by comparing optimisation performances in an unbounded search space and a pruned search space (see

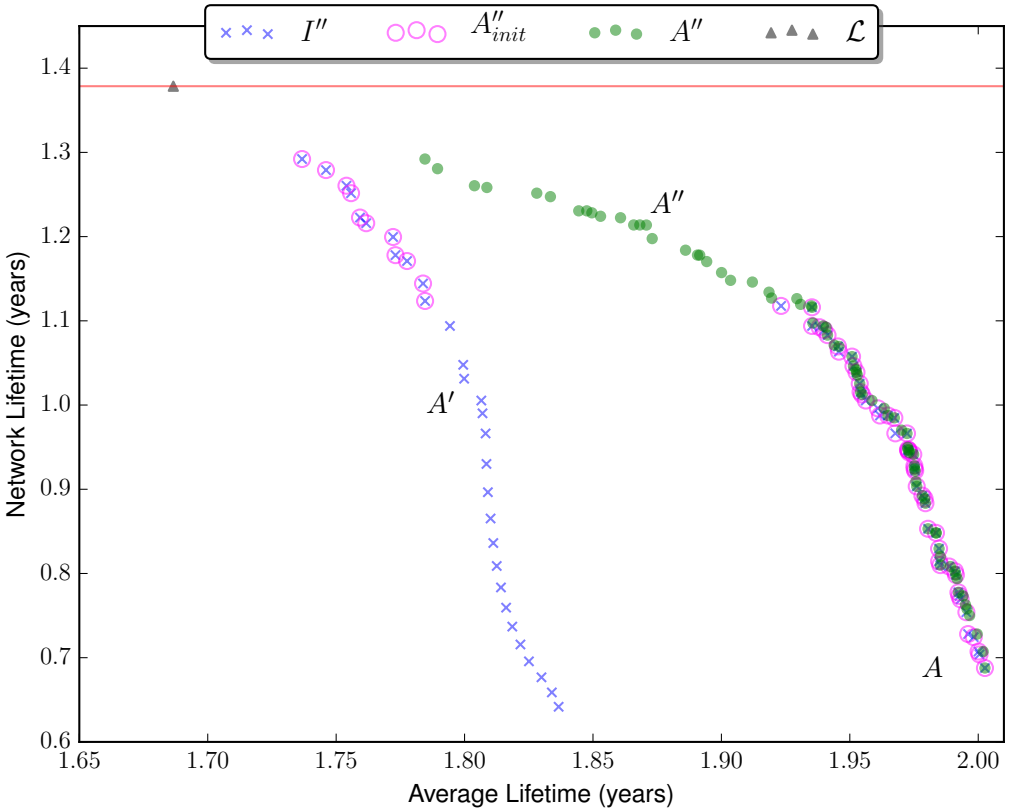


Figure 3.4 Final stage of optimisation. The initial population (represented with crosses) is generated by combining the archives from the previous stage; c.f. Figure 3.3. The non-dominated elements of $A \cup A'$ are ringed in magenta, and form the initial archive of the second stage routing optimisation problem (A''_{init}). Solid green dots show the final approximation of the Pareto set (A''). The maximum minimum lifetime solution if nodes were allowed to use multiple routes rather than a single route is indicated by the grey triangle. The horizontal red line through $\mathcal{L}$ is an upper bound to the network lifetime.

Figures 3.5 and 3.6). In this case, to achieve an approximation of the unbounded search space, we select $k = 10000$ without considering the shortest path in the initial archive and the max-min pruning method. This results in a search space with $10^{3(N-1)}$ solutions, which is significantly larger than the pruned search space with $2 \times 10^{(N-1)}$ solutions. The simulation results indicate that the single-stage approach in the larger space perform poorly in comparison to the single- or two-stage approach in the pruned space with the same number of function evaluations. This confirms the effectiveness of using the search space pruning methods for rapid approximation of optimal routings.

3.5.2 Protecting a Group of Nodes

In practice, it is often desirable to give priority to one or more nodes so that their lifetimes are prolonged, because, for example, they are hard to access. To address this issue, we introduce an additional objective (3.8) with the aim of prolonging the minimum lifetime of the priority group of nodes, alongside aforementioned objectives.

We selected a priority group, $\mathcal{U} = \{v_8, v_{19}, v_{21}, v_{26}\}$ to include two heavily-loaded nodes (v_{19} and v_{21}) and two others. In Figure 3.7, we show the estimated trade-off surface in

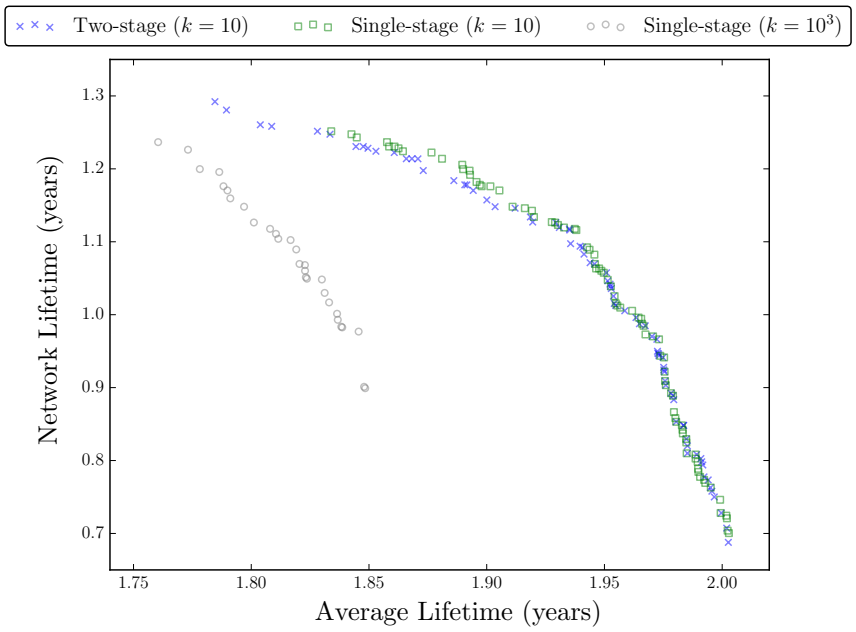


Figure 3.5 Comparison between single-stage optimisation (green squares) and two-stage optimisation (blue crosses) strategies. The single-stage approach performs as well as two-stage approach, except at the extreme network lifetime end. The grey empty dots show the estimated front when the search space is much larger with $k = 10000$ and without max-min pruning, which is significantly worse than approximations made in pruned search space.

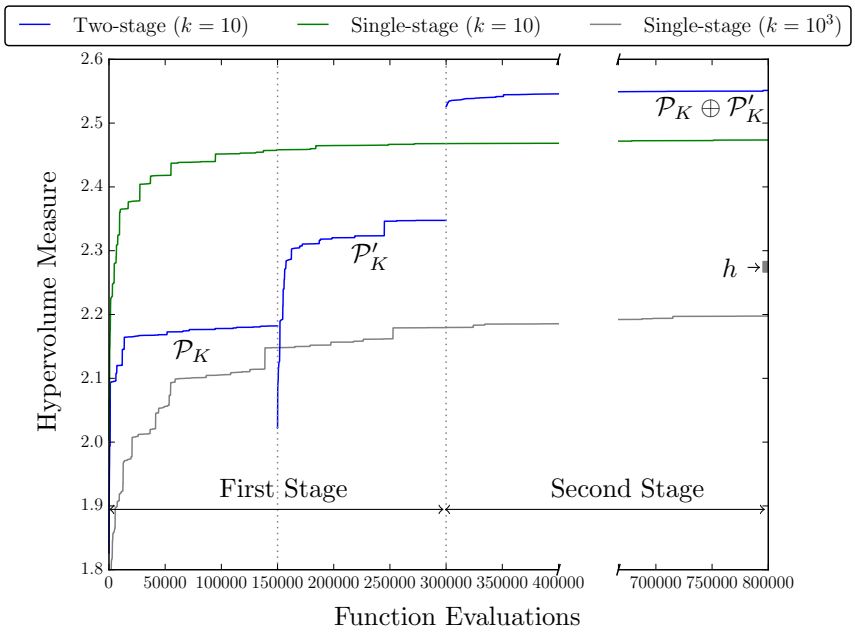


Figure 3.6 Comparison between the hypervolume progressions of two-stage (blue) and single-stage (green) approaches. In the first stage, optimising only in $\mathcal{P}_K$ (0 to 150000) and $\mathcal{P}'_K$ (150000 to 300000) the hypervolume is close to convergence after about 150000 function evaluations. At this point, switching to the combined $\mathcal{P}_K \oplus \mathcal{P}'_K$ provides an immediate improvement in hypervolume over the single-stage hypervolume. The single-stage technique is unable to match the two-stage with the same number of function evaluations, even when optimising using the combined path libraries $\mathcal{P}_K \oplus \mathcal{P}'_K$. The additional grey line depicts the hypervolume progression when the search space is significantly larger with $k = 10000$ and without max-min lifetime pruning for an arbitrary single-stage optimisation run, and h (grey rectangle) shows the hypervolume measure after 2000000 function evaluations. It clearly indicates that if the pruning methods are not used, the optimisation process is likely to substantially underperform with the same number of function evaluations.

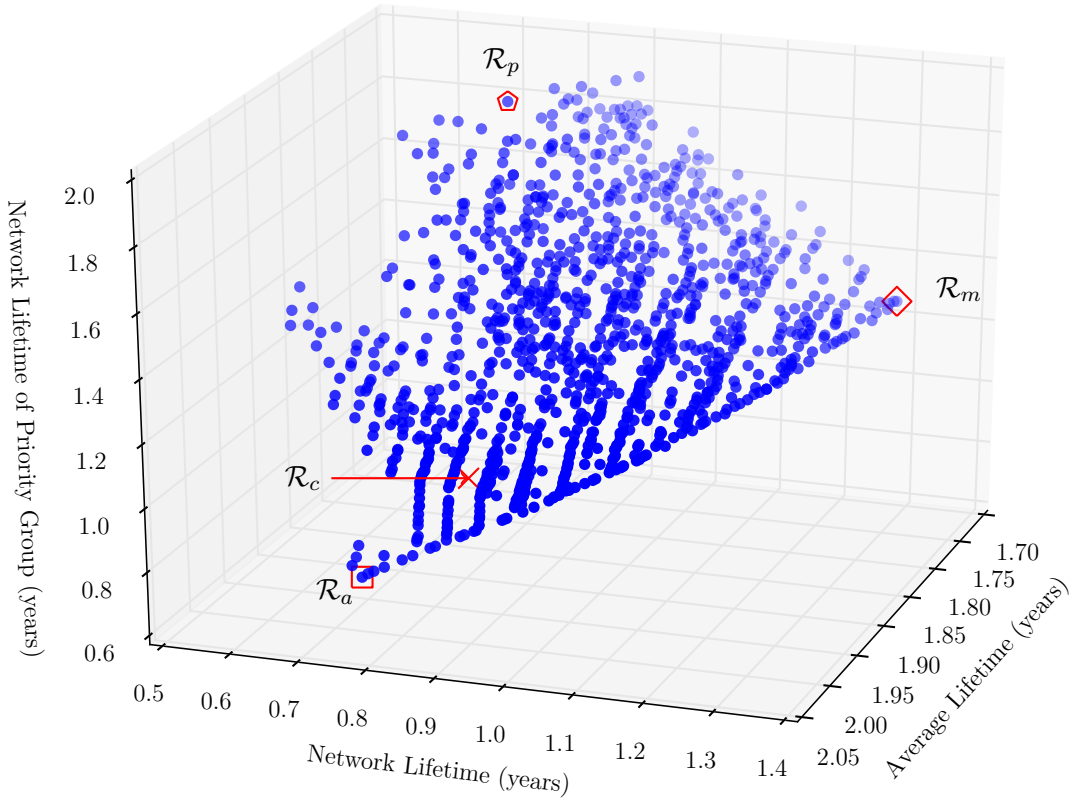


Figure 3.7 Pareto front approximation (blue shaded circles) with three objectives for the Victoria & Albert museum network. The solutions with best average lifetime, $\mathcal{R}_a$, best overall network lifetime, $\mathcal{R}_m$, and best network lifetime of the priority group, $\mathcal{R}_p$, are indicated. The minimum energy solution $\mathcal{R}_c$ is dominated by all solutions in the Pareto front approximation.

3-dimensions between average lifetime, overall network lifetime and network lifetime of a priority group of nodes. Except for the additional objective, the hyper-parameters and general strategy for optimisation remain the same as before.

In Figure 3.8 we show the connectivity map as an adjacency matrix partitioned to place the priority group at the top left. The distinct nature of routing schemes corresponding to the extreme solution with respect to each of the objectives is depicted through the visualisation of the edge utilisations shown in Figure 3.9; here the grey scale indicates the frequency with which an individual radio link (edge) is used in a particular routing scheme.

To illustrate the underlying distinction in nature of the extremal solution for each objective, we consider the edge utilisations, i.e. the number of times a particular edge is used in a routing scheme, of three particular routing schemes: the best average lifetime ($\mathcal{R}_a$), the best overall network lifetime ($\mathcal{R}_m$) and the best network lifetime of priority group ($\mathcal{R}_p$) solutions. In Figure 3.9, we see that in $\mathcal{R}_a$ most paths utilise the link between node v_{19} and the base station v_B. As minimum energy routes are preferred, the average lifetime is high (1.99 years), but overall network lifetime and network lifetime of the priority group are low. The network lifetime is low as nodes on minimum energy paths relay most routes, and as a consequence exhaust their batteries quickly. The overall network lifetime and network lifetime of the priority group are the same, 0.73 years. This is because the most

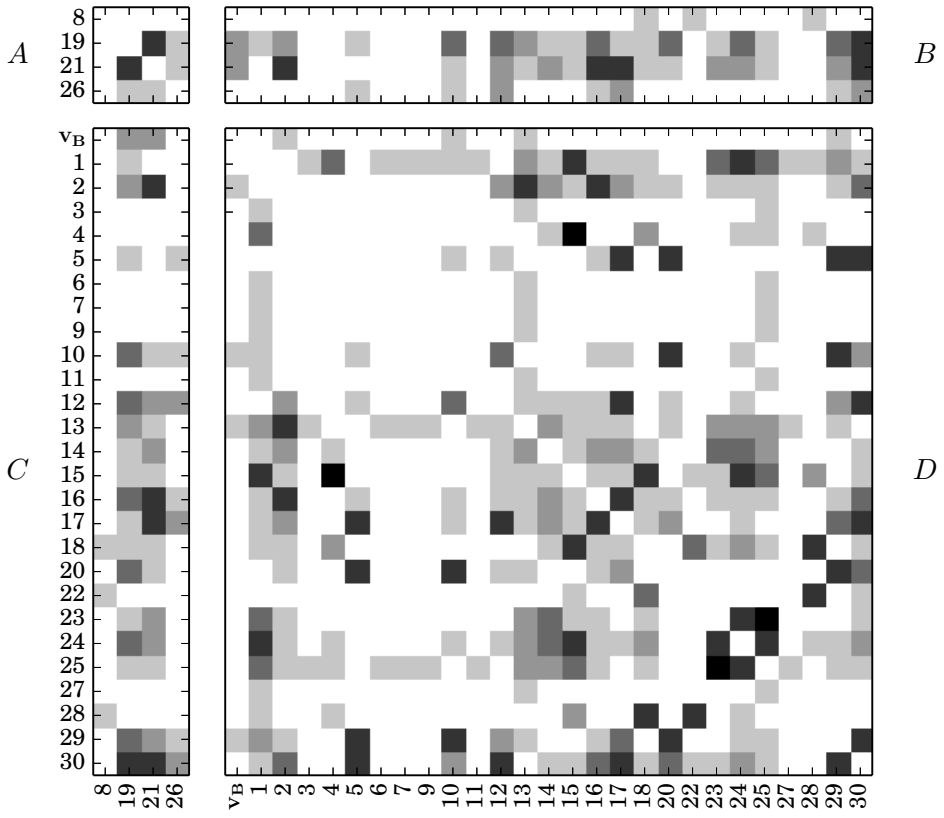


Figure 3.8 Adjacency matrix showing connectivity of nodes in the Victoria & Albert Museum network. The numbers on the left and bottom show the node indices; the base station is v_B. The grey scale indicates the energy consumption required for communication, with darker shades denoting less energy; white cells indicate there is no radio link between cells (infinite energy required for communication). The matrix is partitioned into submatrices showing connectivity between nodes in the priority group (A), nodes outside the priority group (D) and the inter-connectivity between priority and non-priority nodes ($B^\top = C$).

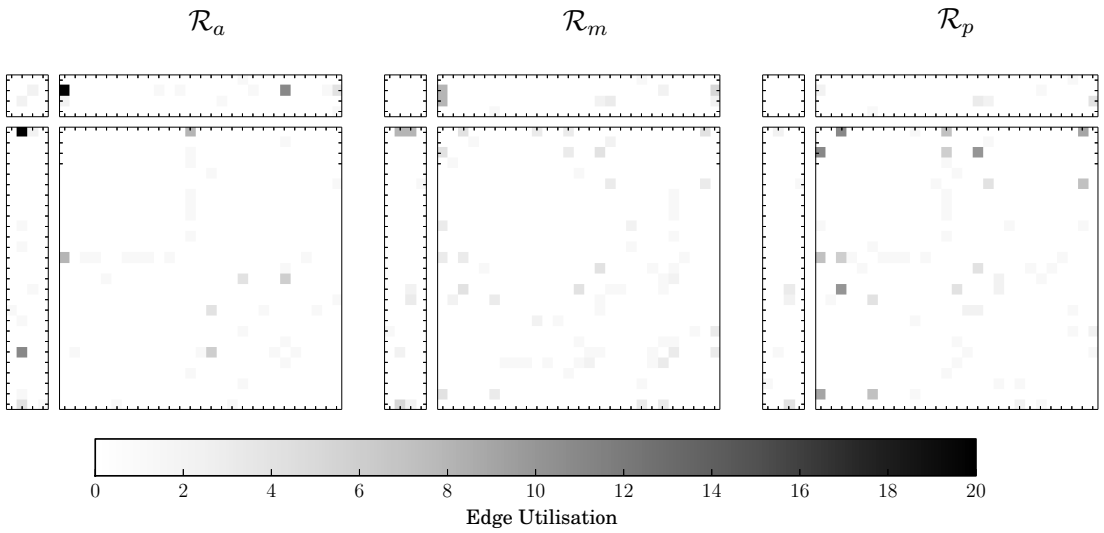


Figure 3.9 Edge utilisations for best average lifetime $\mathcal{R}_a$, best overall minimum lifetime $\mathcal{R}_m$ and best minimum lifetime for the priority group $\mathcal{R}_p$ routing schemes. Higher edge utilisations are represented by darker greys.

energy efficient routes heavily load v_{19}, which is in the priority group, so causing it to exhaust its battery first.

In contrast, $\mathcal{R}_m$ provides a more evenly distributed edge utilisation map, in particular v_{19} and v_{21} share the load between them. In this case, an increase of 78.1% in network lifetime when compared with $\mathcal{R}_a$ is achieved, at the expense of not using all energy efficient routes. Hence, the average lifetime is reduced by 12.5%. The minimum lifetime of the priority group is 1.3 years as v_{19} is the minimum lifetime node. We note that in this case the intra-connectivity between protected group nodes also diminishes.

The most dramatic difference is in $\mathcal{R}_p$, where most paths avoid the priority nodes, improving the minimum lifetime for the priority group by 48.3% for v_{19} with respect to $\mathcal{R}_m$. To achieve this increase in minimum lifetime for the priority group, high energy cost paths must be used for paths in the rest of the network resulting in a reduction in the minimum lifetime for the network as a whole; v_2 is the node whose battery is first exhausted in this case.

3.6 Better Approximation of Maximum Lifetime Routing

It is of considerable practical importance to maximise the minimum lifetime of any node in the network, because the network requires no maintenance until the battery of the shortest-lived node is exhausted. Chang and Tassiulas [2004] described how this can be modelled as an LP problem. The LP formulation relies on each node being able to use many paths to the base station, which in turn enables load balancing by using different routes for fractions of the network lifetime. However, the computational power and the storage requirements associated with switching between many routing schemes can be prohibitive for low-powered devices. Therefore, so far in this thesis, we have only considered a *single-path routing scheme* for the network, where each node has a single path to the base station through the network lifetime, that is the time until any node exhausts its battery. However, we can extend this idea to a *multi-path routing scheme* in which only a small number of routing schemes (two or three) are available to the network, and all nodes change at pre-determined times to a new scheme. This re-routing can be accomplished at low energy cost from the base station. Note that this is a limited version of a general multi-path routing strategy as paths for nodes change together rather than being independent. In this section, we describe an evolutionary algorithm that optimises a small number of routing schemes together, and thus achieves a better approximation to the LP solution quality in comparison to using a single routing scheme throughout the network lifetime.

Intuitively, to see how using multiple routing schemes in a time-shared manner can be useful, suppose that under the initial routing scheme $\mathcal{R}$, a node $v^\star$ has the minimum lifetime. Now suppose that the network is optimised a second time after it has been in operation for a while. The result of this second optimisation can be a routing scheme in which $v^\star$ is very lightly loaded, prolonging its life, while a different node, v', which was

lightly loaded in the first epoch, now carries a heavier load. In this way the life of $v^\star$ is extended and it may be that $v^\star$ or v' or some other node is exhausted first, but in any case the time before any single node is exhausted is delayed.

3.6.1 Optimal Time Share

Inevitably the question now is how to best distribute the time share between multiple routing schemes. For a particular routing scheme, the current drain per reporting cycle is constant (the sum of a quiescent current together with transmission and reception costs to each of the nodes it communicates with). The decay in stored battery charge is therefore linear in time, although the decay rate depends on the particular routing scheme used. With this (good) assumption, the times for which each routing scheme should be used can be found by linear programming.

Consider a single routing scheme. If the initial charge at node v_i is q_{0i}, then after running the system with routing scheme $\mathcal{R}$ for time τ the charge remaining at the node is q_i:

$$q_i = q_{0i} + m_i\tau, \tag{3.16}$$

where, $m_i < 0$ is the rate of decay determined by the particular routing scheme in operation and given by $m_i = -(B_i + C^i)$ (c.f., equations (3.4) and (3.5)).

In a similar fashion, if there are D routing schemes, $\mathcal{R}_1, \ldots, \mathcal{R}_D$, in operation for times $\tau_1, \ldots, \tau_D$, the charge decay equation for v_i becomes:

$$q_i = q_{0i} + \sum_{d=1}^{D} m_{id}\tau_d. \tag{3.17}$$

Figure 3.10 illustrates the decay for v_i for multiple routing schemes.

Denoting the vector of time shares by $\boldsymbol{\tau} = (\tau_1, \tau_2, \ldots, \tau_D)^\top$, the vector of initial battery charges by $\mathbf{q}_0 = (q_{01}, q_{02}, \ldots, q_{0n})^\top$, the vector of remaining charges by $\mathbf{q} = (q_1, q_2, \ldots, q_n)^\top$ and the n by D matrix of decay rates by $\mathbf{M}$, the charge decay equations for all the nodes can be written as:

$$\mathbf{q} = \mathbf{q}_0 + \mathbf{M}\boldsymbol{\tau}. \tag{3.18}$$

The lifetime of the network is $\sum_{d=1}^{D} \tau_d = \mathbf{1}^\top\boldsymbol{\tau}$ subject to the constraint that the charge remaining at all nodes is positive. This allows us to formulate an LP problem for the time shares that will maximise the minimum lifetime of the network, given the D routing schemes as follows:

$$\max(\mathbf{1}^\top\boldsymbol{\tau}) \tag{3.19a}$$

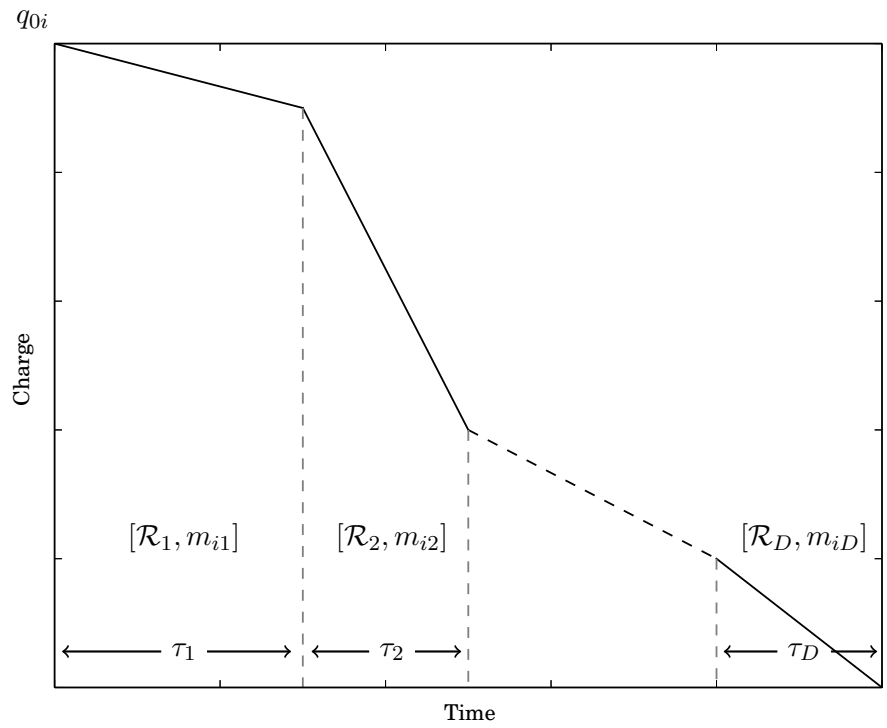


Figure 3.10 Battery charge decay for a node v_i with initial charge q_{0i} using multiple routing schemes. Routing scheme $\mathcal{R}_d$ is active for time span τ_d with a decay slope of m_{id}.

subject to:

$$\tau_d \geq 0 \qquad d = 1, \ldots, D, \tag{3.19b}$$

$$q_i > 0 \qquad i = 1, \ldots, n. \tag{3.19c}$$

Solving this LP problem enables us to derive the optimal time share vector $\boldsymbol{\tau}^*$ for a given set of routing schemes.

The minimum lifetime node will exhaust its battery first. At this point the network would be re-routed for establishing connectivity for all other nodes as they have some charge left. However, discounting the re-routing, we can consider this residual charge at nodes for achieving better overall average lifetime. For such an arrangement, we can assume that the nodes with remaining charge will carry on with the last, i.e. Dth, routing scheme. Hence, the overall average lifetime will depend on the last routing scheme in sequence. The time t_i remaining for node v_i after the minimum lifetime node is exhausted is given by

$$q_{0i} + \sum_{d=1}^{D} m_{id}\tau_d^* + m_{iD}t_i = 0. \tag{3.20}$$

The average lifetime of the network is then

$$\bar{\tau} = \mathbf{1}^\top \boldsymbol{\tau}^* + \frac{1}{N-1}\sum_{i=1}^{N-1} t_i. \tag{3.21}$$

Clearly, although changing the order in which the routing schemes are used does not affect

the network lifetime, the average lifetime depends on which routing scheme is chosen as the last routing scheme, $\mathcal{R}_D$. We therefore select $\mathcal{R}_D$ as the routing scheme that maximises the average lifetime.

Solving this LP problem and finding the routing scheme to use as R_D enables us to achieve the best network lifetime and average lifetime for a given multi-path routing scheme.

3.6.2 Evolving Multi-Path Routing Schemes

In order to locate good routing schemes we consider an evolutionary algorithm in which a solution $\mathcal{R}$ is constructed from D routing schemes: $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_D\}$. In this *D-path* routing scheme approach the evolutionary algorithm is used to optimise the routes comprising each of the constituent schemes, but optimal time shares of these evolved routes are found by solving the LP problem in (3.19). This speeds up the search considerably compared with using an evolutionary algorithm or similar stochastic search to find good time shares as well as the good routing schemes. Note that it is possible for one or more time shares to be zero, so that the evolutionary algorithm can locate solutions in which only a single routing scheme is used.

Despite the pruning used to limit the size of the search space described in section 3.3, the use of several routing schemes leads to a further explosion in search space size because the number of paths available for each node is $2k$ in each of the D routing schemes (using k-shortest path pruning on G and G'). The number of possible solutions is thus approximately $(2k)^{nD}$, although some nodes may have fewer than $2k$ shortest paths available.

In order to combat this combinatorial explosion we adapt the basic evolutionary strategy for a single-path routing scheme to perform a series of evolutionary searches in increasingly large search spaces as follows.

1. Evolutionary searches using path libraries $\mathcal{P}_K$ and $\mathcal{P}'_K$ are carried out separately but evolving single routing schemes rather than multiple schemes. (This can be achieved by replicating a single routing scheme D times and applying the same mutation and crossover operations to all of them.) As illustrated above, optimisation using $\mathcal{P}_K$ and $\mathcal{P}'_K$ separately helps to find good average lifetime solutions and good minimum lifetime solutions by restricting the size of the search space and guiding the search according to the pruning method used.

2. Still optimising using $\mathcal{P}_K$ and $\mathcal{P}'_K$ separately, we seek D-path solutions. Here at each iteration mutation and crossover operations suggest new paths for nodes in one of the D constituent routing schemes, after which the optimal time shares, and thus the minimum and average lifetimes, are calculated by linear programming in (3.19). As we illustrate later, solutions in which the routing schemes are the same throughout the lifetime of the network can produce better average lifetime solutions. Therefore, in half the iterations, a new solution is generated by replicating one of the component routing schemes D times.

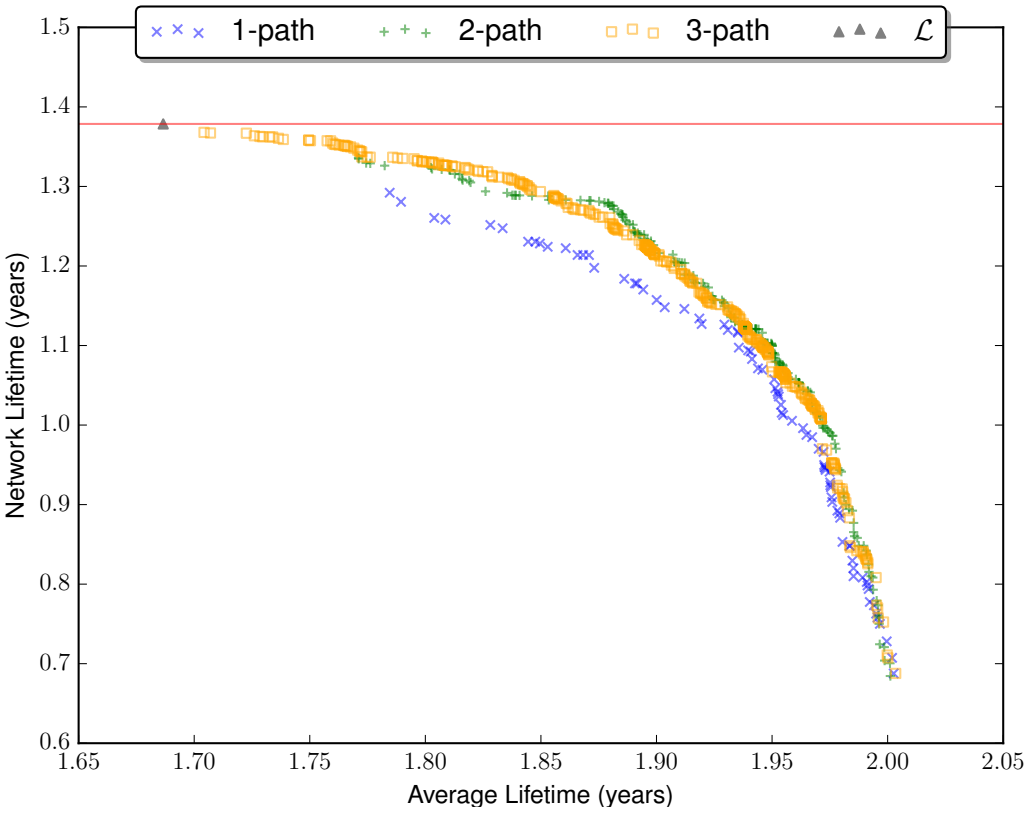


Figure 3.11 Performance comparison between optimisation with 1-path (blue crosses), 2-path (green pluses), and 3-path (orange squares). The performance of the maximum network lifetime solution $\mathcal{L}$ with unlimited paths (∞-path) is shown with a triangle. The horizontal red line through $\mathcal{L}$ shows the upper bound for network lifetime for the given network.

3. Finally, the non-dominated subset of the union of the archives obtained in step 2 is used as the initial archive for a further optimisation using both $\mathcal{P}_K$ and $\mathcal{P}'_K$.

Recall that, as described in section 3.3.1, the path libraries were constructed from the k-shortest paths based on the composite cost. There the edge weights in (3.14) were calculated using the initial battery charges. However, the batteries become depleted by the network operation due to the routing scheme. We therefore, albeit infrequently (e.g. every 100000 function evaluations), update the path libraries with k-shortest paths calculated on the basis of battery charge distributions from partially drained networks, and add any newly discovered paths to the path library. This allows the evolutionary mechanism to choose paths better suited to a network that has been in operation for some time.

3.6.3 Performance Comparison

We compare the performance of optimised 1-path, 2-path and 3-path routing schemes on the Victoria & Albert Museum network.

For both 2-path and 3-path strategies, we used path libraries generated with $k = 10$ shortest paths. The first stage of the optimisation (1-path) was run for 150000 iterations, while the second stage (multi-path routing schemes using $\mathcal{P}_K$ or $\mathcal{P}'_K$ separately) were run for a further 800000 iterations. Finally the third stage (multi-path routing schemes and

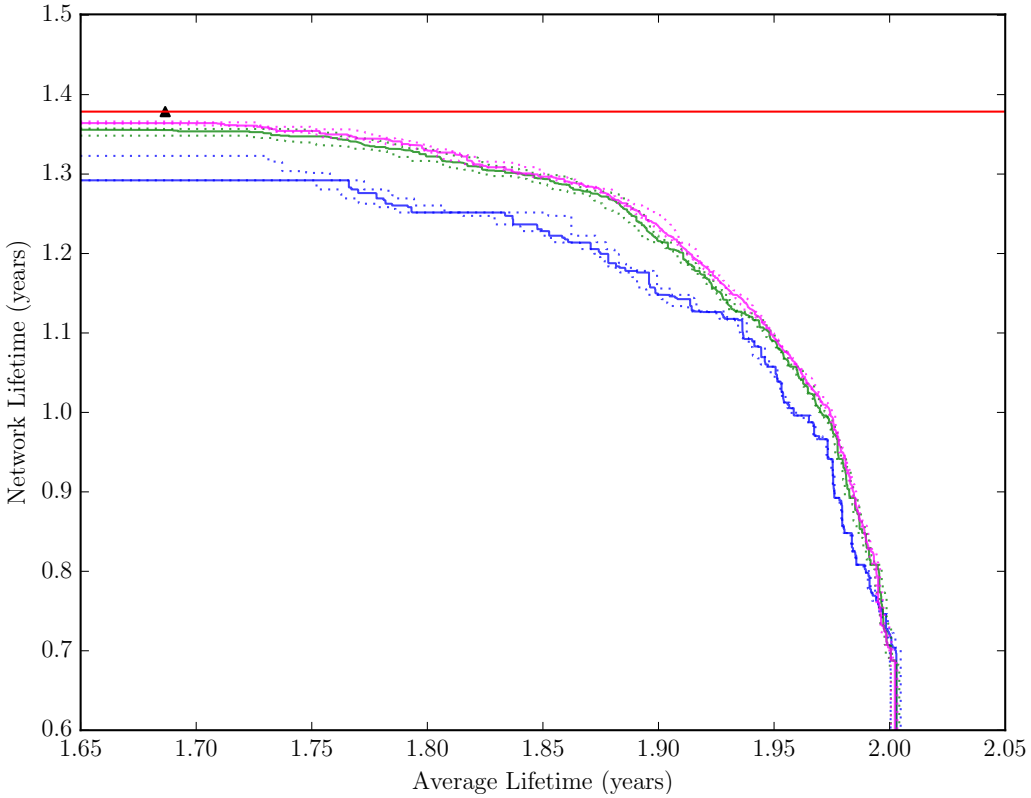


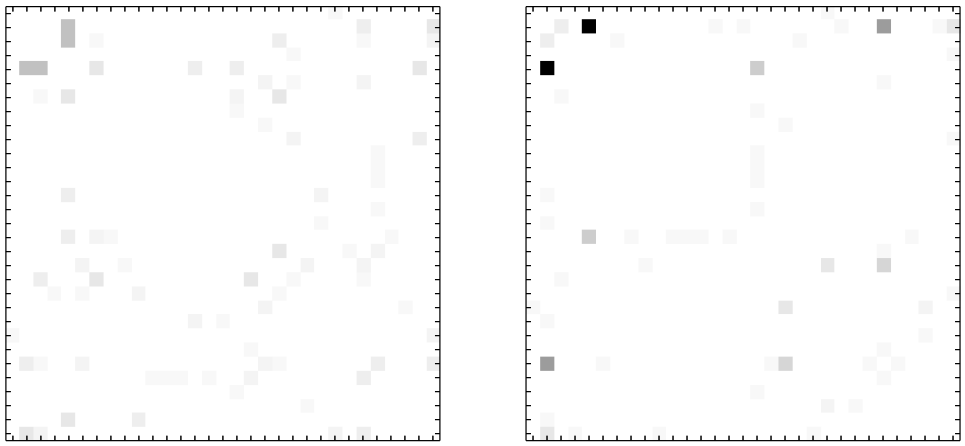
Figure 3.12 Performance comparison between 1-path (blue), 2-path (green), and 3-path (magenta) approaches in terms of summary attainment surfaces over 10 runs with solid lines representing 50% attainment surfaces, and dotted lines depicting 10% and 90% surfaces. The performance of the maximum minimum lifetime solution with unlimited paths is shown with a black triangle. The horizontal red line through $\mathcal{L}$ shows the upper bound for network lifetime.

combined path libraries) was run for an additional 1000000 iterations. These numbers of iterations yield well-converged fronts, and for practical purposes fewer iterations can obtain a good approximation.

In Figure 3.11, we show the estimated fronts for 1-path, 2-path, and 3-path strategies. As may be expected, the estimated Pareto front for 2-path dominates or equals the 1-path scheme and it in turn is equalled or dominated by the 3-path front. The 1-path solution has a maximum network lifetime of 93.7% of the maximum minimum lifetime routing solution, $\mathcal{L}$, which has the freedom to use unlimited routes. The 2-path strategy achieves an improved minimum lifetime which is 97.8% of $\mathcal{L}$. The additional flexibility inherent in the 3-path strategy allows a network lifetime that is 99.2% of $\mathcal{L}$ to be found.

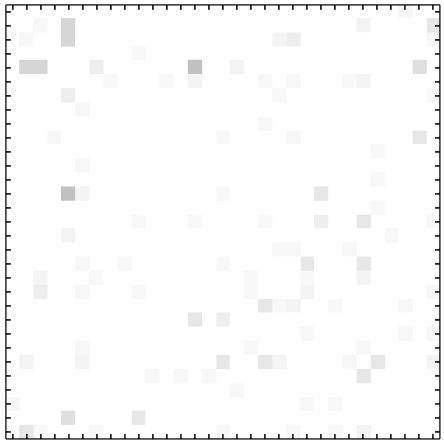
In Figure 3.12 we present the summary attainment surfaces (see section 2.3.4, Fonseca and Fleming [1996], and Knowles [2005]) for 1-path, 2-path, and 3-path strategies calculated over 10 runs. The narrow width between 10% and 90% attainment surfaces clearly indicates the desirable repeatability and convergence of this approach. It also shows that the approximation improves with each additional routing scheme; however, the rate of improvement diminishes as we increase the number of routing schemes.

Edge utilisations for best minimum lifetime and best average lifetime 1-path and 2-path solutions are shown in Figure 3.13. As noted previously, with a single routing scheme

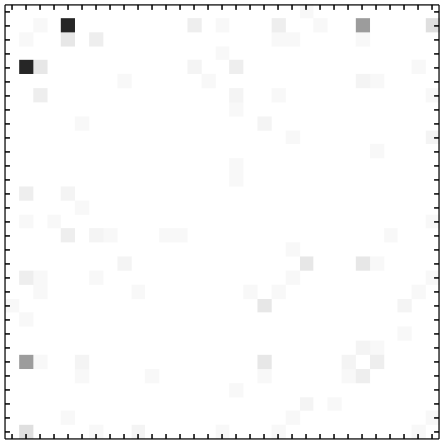


(a) 1-path, maximum network lifetime, R^m

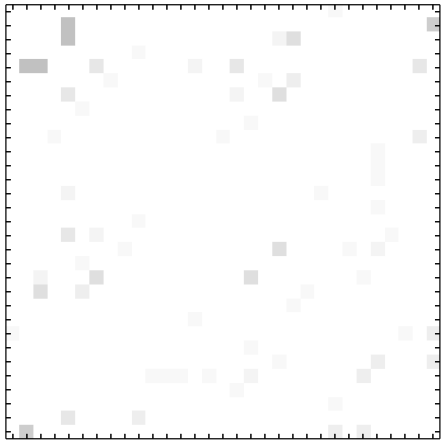
(b) 1-path, maximum average lifetime, R^a



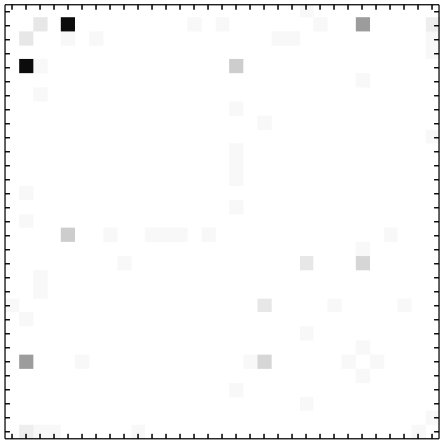
(c) 2-path, max-network lifetime; 1st routing, $R_1^{'m}$



(d) 2-path, max-average lifetime; 1st routing, $R_1^{'a}$



(e) 2-path, max-network lifetime; 2nd routing, $R_2^{'m}$



(f) 2-path, max-average lifetime; 2nd routing, $R_2^{'a}$

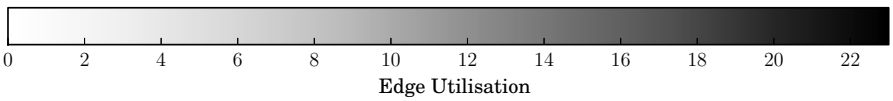


Figure 3.13 Multi-path routing scheme edge utilisations. The top row shows edge utilisations for the best network lifetime R^m and best average lifetime R^a solutions using 1-path routing scheme. Edge utilisations for the two components of the best network lifetime for a 2-path solution shown in (c) and (e) and for the best average lifetime (d) and (f).

(top row of Figure 3.13) the best average lifetime solution heavily utilises a few edges, whereas the load is more evenly distributed for the maximum network lifetime solution. This pattern is also evident in the 2-path solutions. The two component routing schemes, $R_1'^a$ and $R_2'^a$, for the best average lifetime solution are both similar to the 1-path solution. In fact, the 1-path solution has a slightly better average lifetime than the 2-path solution and in general we find that good average lifetime solutions tend to have a single path.

Edge utilisations for the best minimum lifetime 2-path component schemes $R_1'^m$ and $R_2'^m$ are shown in Figures 3.13c and 3.13e. These are active for 0.3 years and 1.05 years respectively, amounting to a total minimum lifetime of 1.35 years. However, individually $R_1'^m$ and $R_2'^m$ can only provide minimum lifetimes of 0.88 and 1.29 years respectively, with the latter being the same as the best minimum lifetime solution from 1-path (R^m in Figure 4.4a). As the different distributions of edge utilisations in Figures 3.13c and 3.13e show, the additional lifetime is achieved by protecting some nodes while the first component is active, but then loading them more while the second component is active. Here these components are active for 22.2% and 77.8% of the network lifetime. The extra flexibility provided by a third component produces a small increase in achievable lifetimes, but the additional component is only used for a small fraction of time: the best minimum lifetime 3-path scheme uses components for 6%, 18.7%, and 75.3% of the network lifetime.

Figure 3.11 and 3.12 also show that with each additional routing scheme in the solution, the approximation of the Pareto front improves. As well as improving the range of minimum lifetime solutions located, it also becomes denser. We attribute this to the additional flexibility to introduce paths that provide a finer grained trade-off between the objectives.

3.7 Summary

Using mesh network topologies in low power wireless sensor networks poses a challenging problem of finding routes that best preserve the lifetime of individual nodes and the network as a whole. In this chapter we have proposed a hybrid multi-objective evolutionary approach to approximate the optimum trade-off between the minimum lifetime for any node and the average lifetime of the whole network, where these objectives are associated with infrequent maintenance and energy efficiency respectively. Although not discussed here, we remark that straightforward re-optimisation from a stored Pareto front approximation is an effective way of re-routing a network in case of node or link failure or if new nodes are introduced [Rahat et al., 2014].

Chang and Tassiulas [2004] have shown that the maximum lifetime routing for a network can be modelled as an LP problem that can be solved in polynomial time to obtain the proportions of messages that should be sent via different routes. However, this formulation requires that each node to be able to use many different paths to the base station. In low-power battery networks this is not feasible due the limited computational power and storage. Hence, we use an evolutionary approach to locate a set of routes for each node that can produce a close approximation to the unlimited-path LP solution (99.2% with

3-path routing scheme).

As a combinatorial problem, the search space size can increase drastically as the number of nodes increases. To reduce this potentially large space while retaining sensible solutions, we use k-shortest path pruning on the full connectivity graph and on a reduced graph. The k-shortest path pruning of the full graph limits the number of routes available to a node and thus limits the search space, while retaining solutions that are energy efficient. On the other hand, the graph reduction method retains paths which tend to yield good network lifetime solutions. Central to the efficacy of our approach is optimisation in these limited search spaces, both for single and multiple routing schemes. In addition, solving exactly for the proportions of time that each component is active in a multiple routing scheme solution obviates the need for an additional, expensive stochastic search.

As we have shown, the flexibility of more than one routing scheme – a multi-path routing scheme – allows better solutions to be located. The nodes in the solutions here are constrained to change to another routing simultaneously. This is a constrained multi-path scheme as routes for nodes are dependent on other nodes' routes for changeover. Ideally, in general multi-path routing schemes, route changes for a node should be independent of other nodes. We investigate this in the next chapter while still restricting the number of routes available to nodes. In this case we propose methods to approximate the trade-off between robustness in data delivery against link or node failure, and network lifetime as it is often more important to focus on network lifetime instead of energy efficiency.

Chapter 4

Robustness and Network Lifetime[1]

In wireless sensor mesh network, each sensor (a node) sends its data to the base station, often by relaying data through other nodes. The routing decisions, for instance considering only the most energy efficient routes, may result in some nodes relaying most nodes' data and thus exhausting their batteries quickly. At this point manual intervention is necessary to change batteries or nodes and restore the network operation. It is therefore more important to extend the time before any node expires – the network lifetime – to minimise maintenance activities. In the previous chapter, we have introduced a multi-path routing approach that enables us to well approximate the optimal routings for network lifetime; however, the multi-paths are constrained, because nodes change their paths together at the same time. In this chapter we remove this dependency, and nodes are allowed to change paths independent of other nodes. In addition, wireless sensors operate in unpredictable and dynamic radio environment, where links between nodes may fail. The usage of such unconstrained multi-paths are capable of battling against link unpredictability if data is sent through routes that share as few links as possible. This works because in an event of link failures in one route, partial data may be retrieved from the affected sensor(s) through other routes. In devising robust multi-paths, energy consumption and loads on a node are not of concern, and as a result robust paths may have deleterious effects on battery life. In this chapter, we therefore propose an evolutionary approach to estimate the optimal trade-off between network lifetime and robustness.

In WSNs, alongside network lifetime, it is important that data from sensors are delivered reliably. Data delivery may fail for various reasons: node failure, link failure, and congestion [Paradis and Han, 2007]. In modern deployments, node failure is a rare event due to recent advances in WSN technology. Moreover, it is considered to be a critical event that requires immediate attention from the network administrator so that faulty nodes may be replaced.Additionally, the congestion problem at the Medium Access Control (MAC) layer may be addressed with Time Division Multiple Access (TDMA) techniques [Cionca et al., 2008]. Furthermore, Forward Error Correction (FEC) methods can be deployed to improve link reliability in the Data Link Layer (DLL) [Jeong and Ee, 2003]. However,

[1]Some of the materials presented in this chapter have been submitted for publication in [Rahat et al., 2015a].

even if these techniques are used, unpredictable link failures due to the changing radio environment may still occur [Cerpa et al., 2005; Zhao and Govindan, 2003].

In the literature, many terms have been used to refer to the fault tolerance in data delivery of a routing scheme against link failures [Akyildiz et al., 2002; Yick et al., 2008; Chen et al., 2009]. The most common are: resilience, reliability and robustness. Resilience is generally used to refer to the ability of a network configuration to recover from failures and restore functionality rapidly by dynamically adjusting its configurations, cf. [Ganesan et al., 2001]. Therefore, this term is appropriate for distributed protocols as they take routing decisions reactively, and thus this term is not suitable in the context of this thesis. The distinction between reliability and robustness is not well-defined in WSNs, and both terms are often used in the same context, see for example [Shin et al., 2007; Yahya and Ben-Othman, 2009]. However, in semantic sense, reliability means achieving a particular performance level, and does not necessarily signify the ability to cope with the uncertainties in the system. In contrast, robustness refers to the ability to continue operating effectively despite unpredictable failures. In this thesis, we focus on maintaining the ability of every node to communicate to a central base station while considering the link failure probabilities. We therefore use *robustness* to refer to the fault tolerance in data delivery. Furthermore, we formulate a novel robustness measure that quantifies the *fragility* of a network configuration. This provides us with an avenue to compare different configurations and thus estimate the optimal routings.

In this chapter, we are interested in locating efficient multi-paths that trade-off between network lifetime and robustness; this multi-objective problem is analytically intractable (we discus this in section 4.2). We therefore propose an evolutionary routing optimisation framework to estimate optimal multi-path routing schemes. This approach can achieve solutions with network lifetimes close to the theoretical maximum network lifetime (when no constraint on the number of routes per node is imposed) as presented in [Chang and Tassiulas, 2004], and a range of solutions representing various levels of robustness.

The main contributions of this chapter are as follows:

- We describe a hybrid evolutionary search procedure to approximate the optimal trade-off between network lifetime and network robustness.
- We introduce a novel robustness measure (the *fragility*) of multi-path routing schemes to quantitatively analyse and compare the robustness of different multi-path routing schemes.
- We show how the proportion of time for which each path should be used in a multi-path scheme may be determined by an appropriate linear program to optimise either network lifetime or robustness.
- In addition to pruning search space with k-shortest paths and max-min lifetime pruning (as discussed in section 3.3), we use novel search space pruning methods, based on braided and edge disjoint paths, which in turn speed the search by restricting the search space to regions likely to contain good (robust) solutions.

- The proposed methods are illustrated in a real network deployed in Victoria & Albert Museum, London, and successfully locate a wide range of robust multi-path routing schemes with long network lifetimes and greater robustness, surpassing the performance of single-path routing schemes.
- We discuss how incorporating vertex enumeration in case of alternative optima may improve local lifetime and fragility for a given multi-path routing scheme.
- We show that an extension to the evolutionary method using weighted sum may improve the quality of the approximation.

The rest of the chapter is structured as follows. In section 4.1 we describe our network model and the associated formulation of network lifetime and robustness. This recapitulates some material from chapter 3, but is included here for completeness. Section 4.2 describes the multi-objective problem to be optimised and in section 4.3 a hybrid evolutionary algorithm to solve it is presented. A new search space pruning using braided and edge-disjoint paths is discussed in section 4.3.1, and illustrations on synthetic and real networks are also given. We also discuss possible extensions to the evolutionary approach that may improve local performances in sections 4.3.4 and 4.3.5. Finally, the chapter summary is presented in section 4.4.

4.1 Network Model, Lifetime and Robustness

In this section, we present a model for WSNs with unconstrained multi-path routing, and formulate network lifetime and robustness as objectives to be optimised. For clarity and completeness, we repeat some of the concepts discussed in chapter 3.

4.1.1 Network Model

We consider a communication protocol in which all nodes periodically (e.g., once every minute) send their sensed data to the base station, potentially by relaying a message through one or more nodes. Such data reporting periods are repeated throughout the network lifetime: the time before which any node exhausts its battery. This scenario is most common in industrial applications, especially for constant monitoring of locations.

Once a connectivity map, showing which nodes may communicate with each other, has been established, routing is performed under the assumption that links are reliable. Generally, pairs of nodes are configured to use the most energy efficient settings that allow reliable communication. Usually energy efficient links correspond to high baud rate and low transmission power.

We deem the hardware to be reliable, and thus node failure is a rare event that necessitates replacement of the node. On the other hand, the radio environment is seldom constant and links may occasionally fail due to changing atmospheric conditions, the passage of people, radio interference, and so on. One mechanism to combat the intermittent failure

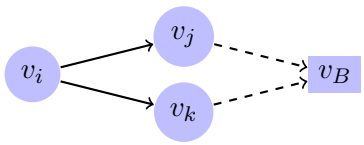


Figure 4.1 Two paths $R_{i1} = \langle v_i, v_j, \ldots, v_B \rangle$ and $R_{i2} = \langle v_i, v_k, \ldots, v_B \rangle$ from v_i to the base station v_B. In a multi-path routing scheme, R_{i1} is active for time share τ_{i1} and R_{i2} is active for time share τ_{i2}.

of links is to provide more than one path from each node to the base station. Each node then splits its traffic between the available paths, sending a proportion of messages via each of the available paths; exactly one path is used on each data reporting cycle, rotating between the available paths. Thus if there is a failure on one path, messages sent via other paths will still be received, providing at least partial information. We call the proportion of time that a particular path is utilised the *active time share* for that path.

Recall that a WSN is represented as a network graph, $G = \{V, E\}$, where V is a set of $N-1$ sensor nodes v_i plus a base station node v_B, and E is the set of M edges, describing with which other nodes each node can communicate [Cormen et al., 2001]. Figure 4.1 illustrates a multi-path routing in which there are two routes from node v_i to the base station v_B: $R_{i1} = \langle v_i, v_j, \ldots, v_B \rangle$ and $R_{i2} = \langle v_i, v_k, \ldots, v_B \rangle$. We denote by τ_{id} the active time share of path R_{id}, namely the proportion of messages sent by v_i via route R_{id}. Clearly, $\tau_{id} \geq 0$ for all i, d and $\sum_d \tau_{id} = 1, \forall v_i \in V$.

Hence, we define a D multi-path routing scheme $(\mathcal{R}, \mathcal{T})$ as a set of paths, where each node has D routes to the base station, and a set of associated time shares:

$$\mathcal{R} = \langle \{R_{1d}\}_{d=1}^{D}, \{R_{2d}\}_{d=1}^{D}, \ldots, \{R_{(N-1)d}\}_{d=1}^{D} \rangle, \tag{4.1}$$

$$\mathcal{T} = \langle \{\tau_{1d}\}_{d=1}^{D}, \{\tau_{2d}\}_{d=1}^{D}, \ldots, \{\tau_{(N-1)d}\}_{d=1}^{D} \rangle. \tag{4.2}$$

Again as discussed in chapter 3, practical memory constraints of the current devices generally require that D, the number of paths for any node, is small, perhaps two or three; this is why the optimal network lifetime as devised by [Chang and Tassiulas, 2004] may not be achieved in a centralised WSN.

4.1.2 Network Lifetime

The energy expended at a node due to transmitting its own data and relaying other nodes' data depletes charge in its battery and thus governs the lifetime of a node. An individual node's lifetime, and thus the network lifetime, then depends on the routing scheme as it dictates the total number of transmissions and receptions involving the node.

Let T_{kj} be the energy (charge) required at node v_k to send a message to v_j and let A_{pk} be the energy required to receive (and acknowledge) a message from v_p at v_k (Figure 4.2). Then in one reporting cycle, the energy cost at v_k associated with relaying messages in

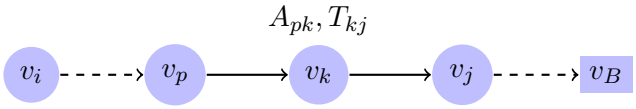


Figure 4.2 Notation for the d-th route for node v_i: $R_{id} = \langle v_i, \ldots, v_p, v_k, v_j, \ldots, v_B \rangle$. The energy costs to send data from v_p to v_k are T_{pk} at node v_p and A_{pk} at node v_k.

path $R_{id} = \langle v_i, \ldots, v_p, v_k, v_j, \ldots, v_B \rangle$ is

$$C_{id}^k = A_{pk} + T_{kj}. \tag{4.3}$$

At the originating node there is no reception cost, so that $C_{id}^i = T_{ir}$ where v_r is the node immediately downstream of v_i.

In order to calculate the battery lifetime remaining due to an unconstrained multi-path routing scheme, we require additional intrinsic information about the nodes, namely the charge q_k remaining in the battery and the quiescent energy consumption per reporting cycle B_k due to constant micro-controller operation, sensor measurements, running an on-board display, etc. The life of the node v_k is therefore given by

$$L_k = \frac{q_k}{N_c(B_k + \sum_{R_{id} \in \mathcal{R}} C_{id}^k \tau_{id})} \tag{4.4}$$

where N_c is the number of reporting cycles per unit time (e.g. one year). Note that the energy expended at node v_k as a result of using route R_{id} depends on the associated time share τ_{id}; more frequent use naturally incurs a greater energy expense. We emphasise that $L_k \equiv L_k(\mathcal{R}, \mathcal{T})$ is a function of all the paths and the associated time shares which utilise v_k.

Note that the formulation for lifetime in equation (3.5) takes into account the combined energy usages at edges due to different routes, and thus it is edge based. In contrast, in equation (4.4) the lifetime formulation considers the energy usages for different routes separately, and therefore it is a route based formulation. Here, as we use an unconstrained multi-paths approach, where nodes may change between available routes without depending on other nodes, we use the route based approach so the contributions due to individual routes may be separated naturally.

Our goal here is to maximise network lifetime, that is the time before any individual node requires its battery to be changed or recharged; thus we seek to maximise

$$L(\mathcal{R}, \mathcal{T}) = \min_{v_k \in V} L_k(\mathcal{R}, \mathcal{T}). \tag{4.5}$$

Optimal Time Share

Suppose that the routes $\mathcal{R}$ have been determined. Then the time shares that best distribute the time share between multiple paths for each node can be found as follows.

Inspecting (4.4), we note that rearranging this equation for a mathematical programming formulation will impose quadratic constraints with variables L_k and τ_{id} in product form, which is an NP-hard problem [Boyd and Vandenberghe, 2004]. In order to reformulate this problem as a linear program, we consider the inverse lifetime

$$L'_k(\mathcal{T}) \equiv 1/L_k = \frac{N_c}{q_k}\left(B_k + \sum_{R_{id}\in R} C^k_{id}\tau_{id}\right) \tag{4.6}$$

so that maximising $\min_{v_k\in V}\{L_k\}$ is equivalent to minimising $\max_{v_k\in V}\{L'_k\}$. Defining $L^{'*} = \max_{v_k\in V}\{L'_k\}$, the optimal time shares are then the solution to the linear program:

$$\min_{\mathcal{T}} L^{'*} \tag{4.7a}$$

subject to:

$$q_k L^{'*} - N_c\left(B_k + \sum_{R_{id}\in R} C^k_{id}\tau_{id}\right) \geq 0 \quad \forall k; \tag{4.7b}$$

$$\tau_{kd} \geq 0 \quad \forall k, d; \tag{4.7c}$$

$$\sum_d \tau_{kd} = 1 \quad \forall k. \tag{4.7d}$$

Note that in deriving the inequality constraints in (4.7b) we utilise the relationship that for all nodes in the system $L^{'*} \geq L'_k$ and replace L'_k by $L^{'*}$ in (4.6). We use this trick because *min-max* functions are non-smooth [Zang, 1980], and considering $L^{'*}$ instead of L'_k lets us formulate a convex linear program which can be solved in polynomial time [Bertsimas and Tsitsiklis, 1997].

Here, the inequalities (4.7b) (derived from (4.4) and (4.6)) ensure that the batteries have non-negative charge, while (4.7d) ensures that every node has paths allocated for all its messages.

Clearly, by solving the linear program (4.7) we can obtain a set of optimal time shares once a multi-path routing scheme is generated. This provides a means to efficiently calculate optimal time shares and thus compare routes $\mathcal{R}$ and $\mathcal{R}'$. In the evolutionary optimisation method that we present below, candidate paths are generated by the stochastic evolutionary mechanism; optimal network lifetimes for these are obtained by solving the linear program, obviating the need to perform a further stochastic search to locate optimal time shares.

4.1.3 Robustness

A network can be considered as robust when it can deliver data despite link failures. Clearly, in a single-path routing scheme failure of a link will cut off any node whose path uses that link. Multi-path routing schemes offer the possibility of partial data delivery

during link failures and we therefore seek to characterise the robustness of a multi-path routing scheme to link failures. We consider the expected message loss associated with the failure of a link in one of the paths used by a single node. We then characterise the *fragility* of the network as the maximum expected data loss in the event of a link failure anywhere in the network. Then maximising robustness is equivalent to minimising the fragility for a given multi-path routing scheme.

Let π_m be the failure probability of an edge $e_m \in E$. We assume that the edge failure probabilities are independent, so that the probability that a path $R_{id} \in \mathcal{R}$ fails in a unit time can be written as:

$$p_{id} = P(R_{id} \text{ fails}) = 1 - \prod_{e_m \in R_{id}} (1 - \pi_m) = \sum_{e_m \in R_{id}} \pi_m - \tilde{p}_{id}, \tag{4.8}$$

where $\tilde{p}_{id}$ represents all the higher order product terms. When the edge failure probabilities are small ($\pi_m \ll 1$), the higher order product terms (representing the probability of more than one edge failing) are negligible, and p_{id} is thus well approximated by:

$$p_{id} \approx \sum_{e_m \in R_{id}} \pi_m. \tag{4.9}$$

In the Victoria & Albert network $\pi_m \approx 1\%$ and we examine the effects of this approximation below.

If a node v_i sends U_i messages per unit time, then it uses R_{id} to send $U_i \tau_{id}$ of these messages and the expected loss of messages associated with a failure of a link in R_{id} is

$$U_i \tau_{id} p_{id}. \tag{4.10}$$

Now, in multi-path routing schemes the paths from a particular node may share edges. As a result, failure in a link in R_{id} may also be associated with loss in another path $R_{xy} \in \mathcal{R} \setminus R_{id}$ that shares edges with R_{id}. In this case the expected data loss associated with the failure of an edge is:

$$F_{id} = U_i \tau_{id} p_{id} + \sum_{\substack{R_{xy} \in (\mathcal{R} \setminus R_{id}) \\ v_x \in V}} U_x \tau_{xy} \left(1 - \prod_{e_n \in (R_{xy} \cap R_{id})} (1 - \pi_n) \right) \tag{4.11}$$

where the bracketed term is the probability that an edge common to R_{id} and another path R_{xy} fails.

In the following we use the first order approximation of path failure probabilities:

$$F_{id} = U_i \tau_{id} \sum_{e_m \in R_{id}} \pi_m + \sum_{\substack{R_{xy} \in (\mathcal{R} \setminus R_{id}) \\ v_x \in V}} U_x \tau_{xy} \left(\sum_{e_n \in (R_{xy} \cap R_{id})} \pi_n \right). \tag{4.12}$$

We call $F_i = \max_d F_{id}$ the *fragility* of the multi-path routes for v_i. To quantify the

robustness of the entire network, we define the *fragility of the network* as the maximum expected data loss for any node:

$$F = \max_{v_i \in V} F_i = \max_{v_i \in V} \max_{d} F_{id}. \tag{4.13}$$

Clearly $F \equiv F(\mathcal{R}, \mathcal{T})$ depends on the choice of paths and time shares. To maximise the robustness of the network, we therefore minimise the fragility with respect to $\mathcal{R}$ and $\mathcal{T}$, that is, we seek to minimise the maximum expected data loss in the event of a link failure.

Our work on robustness is complementary to the work presented in [Cerpa et al., 2005; Zhao and Govindan, 2003]: they devise ways to calculate statistical and empirical edge failure probabilities π_m and such information can be used to estimate robustness of multi-path routing schemes. If this information is not readily available, especially when the network is being established, it may be assumed that all edges are equally likely to fail and we set $\pi_m = \pi = const.$ for all m.

We next show that the optimum time shares $\mathcal{T}$ may be found by solving a linear program when the routes $\mathcal{R}$ are known, after which, in section 4.1.3, we illustrate the fragility measure for some simple intuitive cases. In section 4.2 we then present the multi-objective optimisation problem and an evolutionary approach to maximise both the network lifetime and robustness.

Optimal Time Share

In a similar manner to the network lifetime problem, we can devise a linear program to locate the active time shares $\mathcal{T}$ for a particular multi-path routing scheme $\mathcal{R}$ that minimises the network fragility (4.13).

With $F^*(\mathcal{T}) = \max_{R_{id} \in \mathcal{R}} F_{id}$, the linear program to minimise the network fragility can be described as follows:

$$\min_{\mathcal{T}} F^* \tag{4.14a}$$

subject to:

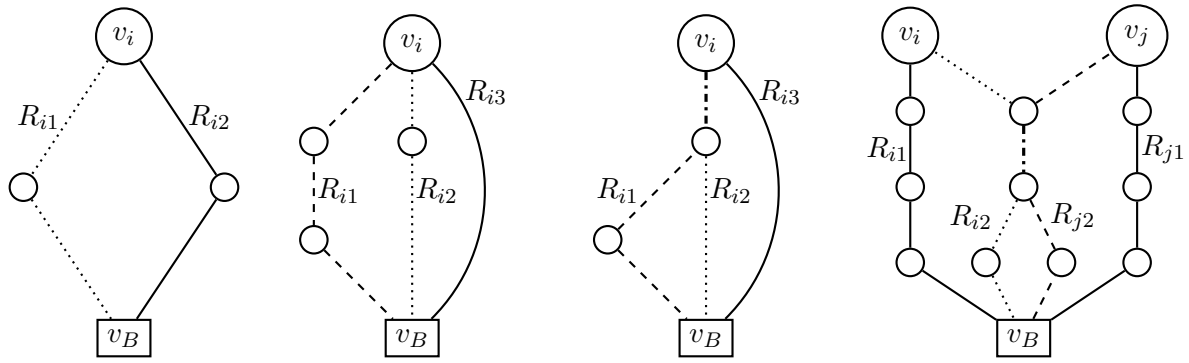
$$F^* \geq F_{id}, \quad \forall R_{id} \in \mathcal{R}; \tag{4.14b}$$

$$\tau_{kd} \geq 0, \quad \forall k, d; \tag{4.14c}$$

$$\sum_{d} \tau_{kd} = 1, \quad \forall k. \tag{4.14d}$$

Here too, setting $F^* = \max_{R_{id} \in \mathcal{R}} F_{id} \geq F_{id}$ enables us avoid the non-smoothness of the min-max problem (4.14a) and formulate the linear program. The set of constraints in (4.14b) indicates that any path specific fragility in (4.11) is less than or equal to the network fragility F^*. As before, the constraints in (4.14d) ensure that all data reporting cycles are used.

Solving this linear program results in a set of active time shares according to the best



(a) Node v_i has two routes to the base station v_B: R_{i1} (dotted), and R_{i2} (solid) with associated time shares τ_{i1} and τ_{i2} respectively. With equal link failure probabilities, the routes are used equally, $\tau_{i1} = \tau_{i2} = 50\%$.

(b) Node v_i has three routes to the base station: R_{i1} (dashed) and R_{i2} (dotted) and R_{i3} (solid). With equal link failure probabilities π, the time shares are $\tau_{i1} = 18.2\%$, $\tau_{i2} = 27.3\%$ and $\tau_{i3} = 54.5\%$ and the fragility is $F_i \approx 0.545 U_i \pi$.

(c) Same configuration as Figure 4.3b except for the link (dash-dotted) shared between R_{i1} (dashed) and R_{i2} (dotted). With equal link failure probabilities the time shares are $\tau_{i1} = 12.5\%, \tau_{i2} = 25\%, \tau_{i3} = 62.5\%$. The fragility is $F_i = 0.625 U_i \pi$.

(d) v_i and v_j each have two routes with equal numbers of links, but share a single link (dash-dotted). With equal link failure probabilities π and traffic $U_i = U_j = U$, the optimal time shares are $\tau_{i1} = \tau_{j1} = 55.6\%$, $\tau_{i2} = \tau_{j2} = 44.4\%$. The fragility of v_i and v_j is $2.22 U \pi$.

Figure 4.3 Case studies illustrating the fragility (4.13). The first order approximation of the route failure probabilities (4.9) is used in expressions for F_{id} (4.12).

robustness for a routing scheme $\mathcal{R}$. This allows us to simply compare robustness of different multi-path routing schemes $\mathcal{R}$ with the knowledge that the active times for each route are optimal, and thus locate routing schemes with better robustness.

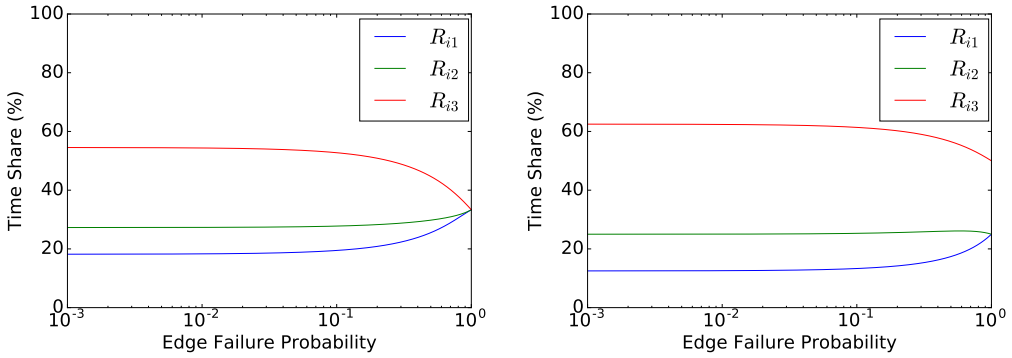
Case Studies

Here we demonstrate the fragility in some simple scenarios, which are illustrated in Figure 4.3.

Edge-disjoint routes between a pair of nodes

Edge-disjoint routes do not share any edges; however, they may share nodes. In such situations, the node fragility F_i is independent of the other nodes and the second term in (4.11) is zero.

As illustrated in Figure 4.3a, node v_i has two routes of equal length to the base station v_B: R_{i1} and R_{i2}. The active time shares for these routes are τ_{i1} and τ_{i2}. If the edge failure probabilities are equal for all the edges, by either solving the linear program or straightforward direct calculation, the minimum fragility occurs when $\tau_{i1} = \tau_{i2}$ and $F_{i1} = F_{i2}$, as would be expected from symmetry. Clearly, if the two routes have equal probability of failing then they should be used equally. Likewise, if the total failure probability for path R_{id} is p_{id} (c.f. (4.9)), then the minimum fragility occurs when the expected losses of



(a) Edge disjoint routes (network graph in Figure 4.3b)

(b) Shared edges between routes (network graph in Figure 4.3c)

Figure 4.4 Effects on optimal time shares among different routes with changes in homogeneous failure probabilities at edges considering the fragility definition in equation (4.8) that incorporates the product terms $\tilde{p}_{id}$ of failure probabilities. For small values of edge failure probabilities, the impact of $\tilde{p}_{id}$ is negligible.

each of the independent routes are equal, which occurs when:

$$\tau_{id} = \frac{1}{p_{id}} \left(\sum_{l} \frac{1}{p_{il}} \right)^{-1}. \tag{4.15}$$

When there are just two routes (Figure 4.3a) then (4.15) simplifies to $\tau_{i1} = p_{i2}/(p_{i1}+p_{i2}) = 1 - \tau_{i2}$. The fragility is $F_i = U_i \left(p_{i1}^{-1} + p_{i2}^{-1}\right)^{-1}$, which is clearly minimised when $p_{i1} = p_{i2}$ so that the risk of failure is borne equally by the two routes. The case for three routes with 3, 2, and 1 links respectively, each with equal failure probability π, is shown in Figure 4.3b. Here the routes are utilised in the proportions $\tau_{i1} = 18.2\%$, $\tau_{i2} = 27.3\%$ and $\tau_{i3} = 54.5\%$ so that the expected loss for each route F_{id} is equalised using the first order approximation to F_{id} in (4.12); at the optimum $F_i = 6U_i\pi/11 \approx 0.545U_i\pi$.

Shared edges from a single node

Sometimes the paths from a single node to the base station will have to share edges. Clearly, this makes the node more fragile because failure of one of the shared links will compromise more than one path, although there might be another independent route.

Figure 4.3c shows a node v_i with three routes in a similar configuration to Figure 4.3b, except that R_{i1} and R_{i2} share a link (shown dash-dotted). With equal link failure probabilities there is less advantage in using R_{i1} and R_{i2} than there was in the previous case. This is reflected in the time shares: $\tau_{i1} = 12.5\%, \tau_{i2} = 25\%$ and $\tau_{i3} = 62.5\%$ and the greater fragility $F_i = 0.625U_i\pi$.

When the failure probability π_m is the same for all edges, i.e. $\pi_m = \pi, \forall e_m \in E$, and the first order approximation of route failure probability p_{id} in (4.9) is used, then the fragility is proportional to π and the precise value of π is not important. When the failure probabilities at edges are large, however, the chance of more than one link failure

is appreciable and the full expression for p_{id} in (4.8) should be used. Figure 4.4 shows the result of using complete expression for p_{id} for network shown in Figure 4.3b and 4.3c. As the figure shows, the higher order terms can be safely ignored when $\pi \lesssim 0.2$, although for routes with many edges this may be lower.

Note that when all links are completely reliable, that is all edge failure probabilities are zero, there are no fragility at the routes, and thus any combination of time shares can be considered as optimal. Similar arguments can be made for the case when all links fail all the time and no data can possibly be delivered.

Multiple nodes sharing edges

When multiple routes share edges, failure of a single link may affect more than one node. In practice, since most nodes in a network generate data, there will be a large number of shared edges and the second term in (4.11) and (4.12) is significant.

As a simple illustration, Figure 4.3d shows two data reporting nodes v_i and v_j, which send their data to the base station each using two paths. These routes have equal numbers of links, so with equal edge failure probabilities, the fragility of each individual route is the same. In this illustration routes R_{i2} and R_{j2} share a single link. In this case, assuming the v_i and v_j generate equal traffic $U_i = U_j = U$ and assuming $\pi \ll 1$, the optimal time shares are $\tau_{i1} = \tau_{j1} = 55.6\%$ and $\tau_{i2} = \tau_{j2} = 44.4\%$, so that traffic is directed away from the shared edge. Without the shared link the time shares for all routes would be 50% and the fragility $2U\pi$, whereas the shared link increases the fragility of both nodes to $2.22U\pi$.

The fragility measure described here may be used for other similar problems. For instance, it may be useful in reducing congestion in transport networks. Generally, such problems are modelled with the predefined edge capacities and the appropriate path flows are sought to reduce the *edge based* congestion factors. Banner and Orda [2007] have described a multi-path scheme to solve this problem. However, if the problem is required to be *route based*, that is the overall congestion in the network due to many commodities being transported through multi-paths are minimised, then the fragility measure may be adapted to achieve this. The modifications required would incorporate considering the reciprocal of edge capacity as equivalent to the edge failure probabilities, the path flows as equivalent to the time shares, and the overall network congestion at routes as equivalent to the fragility. With these assumptions, minimising the overall network congestion would be equivalent to minimising the fragility.

4.2 Multi-Objective Optimisation Problem

Our overall goal is to discover routes and time shares for a network that maximise the network lifetime and maximise the network robustness by minimising the fragility. These two objectives are expressed by (4.5) and (4.13) and may be collected together as a two-

objective optimisation problem:

$$\text{maximise } f_1(\mathcal{R}, \mathcal{T}) = \min_{v_k \in V} L_k(\mathcal{R}, \mathcal{T}), \tag{4.16}$$

$$\text{minimise } f_2(\mathcal{R}, \mathcal{T}) = \max_{R_{id} \in \mathcal{R}} F_{id}(\mathcal{R}, \mathcal{T}). \tag{4.17}$$

These objectives might be augmented with others. For example, as discussed in section 3.5.2, it might be important to maximise the lifetime of one or more particularly inaccessible nodes or to ensure the maximum robustness for other nodes. The optimisation procedure we describe below is easily generalised to these situations, but for simplicity we illustrate our approach with just these two objectives.

We emphasise that in some senses the time shares $\mathcal{T}$ are subsidiary to the routes $\mathcal{R}$, because if $\mathcal{R}$ is known then the $\mathcal{T}$ maximising the lifetime or minimising the fragility may be found by solving the appropriate linear program. For a routing scheme $\mathcal{R}$ we denote by $\mathcal{T}_L(\mathcal{R})$ and $\mathcal{T}_F(\mathcal{R})$ the solutions to the linear programs for lifetime and fragility respectively.

The network lifetime objective in (4.16) ensures that although energy efficient paths are selected, some load is distributed away from the heavily loaded nodes to prolong minimum lifetime. This may be in conflict, however, with the fragility objective in (4.17), where the load distribution that minimises fragility is purely dependent on the nature of the paths and shared edges between paths, without any regard to the traffic carried by each node and therefore the load imposed on its battery. As it will usually be impossible to optimise both objectives with a single routing, we therefore seek the set of solutions corresponding to the optimal trade-off between these objectives.

Just as we described in section 3.1, locating the optimal Pareto set would require searching all possible multi-path routing schemes permitted by the network graph. As counting the number of all possible simple paths in a graph is $\#P$-complete, the corresponding problem of listing all simple paths in order to build multi-path schemes is NP-complete [Valiant, 1979a,b; Roberts and Kroese, 2007]. Therefore, this multi-objective problem is analytically intractable, and hence, we use a hybrid evolutionary approach to approximate the optimal routings.

4.3 Hybrid Evolutionary Approach to Multi-Path Routing Optimisation

We use a straightforward elitist evolutionary algorithm as the basis of our search. The algorithm (see Algorithm 4.1) maintains an archive A of mutually non-dominated $(\mathcal{R}, \mathcal{T})$ solutions which at any stage of the algorithm is the best approximation to the Pareto set.

The algorithm is initialised by constructing s candidate routing schemes from a library of possible routes for each node, $\mathcal{P} = \langle \{R_{1m}\}_{m=1}^{M}, \{R_{2m}\}_{m=1}^{M}, \ldots, \{R_{Nm}\}_{m=1}^{M} \rangle$, where M is the number of routes available for each node.[2] To obtain an efficient search and combat

[2] In practice some nodes may have fewer than M routes available.

Algorithm 4.1 Multi-objective multi-path routing optimisation using evolutionary algorithms.

Inputs

$\mathcal{P}$: Library of paths for each node
T : Number of iterations
s : Size of initial archive
μ : Perturbation rate
c : Crossover rate

Steps

1: $A \leftarrow \text{InitialiseArchive}(\mathcal{P}, s)$ ▷ Initialise random archive
2: **for** $i = 1 \rightarrow T$ **do**
3: $\{\mathcal{R}^1, \mathcal{R}^2\} \leftarrow \text{Select}(A)$ ▷ Select two parent solutions
4: $\mathcal{R}' \leftarrow \text{Crossover}(\mathcal{R}^1, \mathcal{R}^2, c)$
5: $\mathcal{R}'' \leftarrow \text{Perturb}(\mathcal{R}', \mu, \mathcal{P})$ ▷ Mutation
6: $A \leftarrow \text{NonDominated}(A \cup \{(\mathcal{R}'', \mathcal{T}_L(\mathcal{R}'')), (\mathcal{R}'', \mathcal{T}_F(\mathcal{R}''))\})$
7: **end for**
8: **return** A ▷ Approximation of the Pareto set

the combinatorial explosion in the number of possible solutions with increasing network size, this library is built to contain routes that are likely to be good candidates for optimal solutions; details of its construction are given in section 4.3.1. In a D-path routing scheme, D paths for each node v_i are drawn at random from the available routes for v_i in $\mathcal{P}$ to form $\mathcal{R}$. Optimal time shares for $\mathcal{T}_L(\mathcal{R})$ and $\mathcal{T}_F(\mathcal{R})$ are found via linear programming and the candidate solutions $(\mathcal{R}, \mathcal{T}_L(\mathcal{R}))$ and $(\mathcal{R}, \mathcal{T}_F(\mathcal{R}))$ added to A. Solutions in A which are dominated by other solutions are removed from A, so that it is a set of mutually non-dominating solutions.

At each generation of the evolutionary procedure, the routes $\mathcal{R}^1$ and $\mathcal{R}^2$ corresponding to two solutions in A are selected at random from the elite archive A (line 3). These parent routing schemes are combined in a uniform crossover operation (line 4), in which a new routing scheme $\mathcal{R}'$ is constructed by selecting each path R_{id} in the paths for each node v_i from either R^1_{id} or R^2_{id} with probability c or $1 - c$ respectively, independently of other nodes and the paths for v_i. The new routing scheme $\mathcal{R}'$ is then perturbed by choosing a number of paths in the solution to alter based on the perturbation rate μ, and then replacing these from $\mathcal{P}$ at random (line 5). Finally, having evaluated the optimal time shares and the corresponding objectives, if either of the newly constructed routings $(\mathcal{R}'', \mathcal{T}_L(\mathcal{R}''))$ or $(\mathcal{R}'', \mathcal{T}_F(\mathcal{R}''))$ is not dominated by any of the solutions in A, then it is added to A and any solutions in A that are dominated by these are removed from A. In this way non-dominated routing schemes are retained in the archive and the corresponding objectives can only approach the true Pareto front. The evolutionary process continues for a fixed number of generations, although another termination condition, such as a specified minimum dominated hypervolume [Zitzler, 1999] may be employed.

Note that, alongside being a single stage optimiser, the generic framework of algorithm 4.1 differs from algorithm 3.1 in step 6, where appropriate LPs are solved to generate two candidate multi-path schemes and compared with all solutions in the archive.

4.3.1 Search Space Pruning

As noted above, the multi-objective optimisation problem is a combinatorial optimisation problem with, for practical WSNs, a vast number of potential solutions. The number of possible multi-path routing schemes depends on the number of available routes for each node in the network and the number of routes allowed per node. For instance, let the number of available loopless paths from v_i to v_B be a_i. If the number of paths per node is D, then the number of possible multi-path routing schemes is $(\prod_i a_i)^D$. Hence, it is crucial for practical implementations that we derive an efficient algorithm by sensibly pruning the search space, while retaining important potential solutions, rather than considering solutions from the whole search space. In addition to the pruning methods (k-shortest path and max-min lifetime pruning) described in section 3.3, in this section, we describe new methods of pruning the search space: *braided* and *edge-disjoint path pruning* are used to construct libraries of potential paths for each node $\mathcal{P}_{braid}$, and $\mathcal{P}_{edge}$ from the connectivity graph, G; we also reduce G to a new graph G' using *max-min pruning* and construct additional libraries $\mathcal{P}'_{braid}$, and $\mathcal{P}'_{edge}$ from it. We also use the path libraries $\mathcal{P}_K$ and $\mathcal{P}'_K$ generated using k-shortest path pruning on both G and G' respectively. The evolutionary algorithm (Algorithm 4.1) then selects candidate routes from $\mathcal{P} = \mathcal{P}_K \oplus \mathcal{P}_{braid} \oplus \mathcal{P}_{edge} \oplus \mathcal{P}'_K \oplus \mathcal{P}'_{braid} \oplus \mathcal{P}'_{edge}$.

Braided and Edge-Disjoint Path Pruning

In the interest of retaining potential good routes from the perspective of robustness, we build two additional path libraries: $\mathcal{P}_{braid}$, containing braided paths, and $\mathcal{P}_{edge}$, containing edge-disjoint paths, which have been shown to be highly resilient and energy efficient [Ganesan et al., 2001].

Ganesan et al. [2001] describe braided paths as partially disjoint paths, i.e. paths in a braid are allowed to share nodes and edges as long as they differ in some edges or nodes. These braided paths are based on a primary path, which we take to be the shortest path from a node to the base station considering the composite cost as edge weights; these primary paths are found during the construction of $\mathcal{P}_K$ or can be found using Dijkstra's algorithm [Dijkstra, 1959].

Two particular types of braided paths are presented by Ganesan *et al.*: idealised and localised braids. In idealised braids, two paths must differ in at least one node; while in localised braids two paths are allowed to share nodes as long as they have different edges to and from at least one shared node. Therefore to produce idealised braids, we remove one node at a time from the primary path from the network graph and calculate new primary path. Similarly, for localised braids, we remove only the edges connecting a node in the primary path, and generate the primary path on the new graph. This is repeated for each node and associated edges in the primary path. Hence, the number of braided paths depends on the length of the primary path.

To generate edge-disjoint paths, starting from the primary path, we remove edges of the

1-st to m-th path from the network and calculate the next shortest path, i.e. the $(m+1)$-th completely edge-disjoint path, based on the composite edge costs. We repeat this process to construct $\mathcal{P}_{edge}$ with k edge-disjoint paths. In a partially connected network, as is usually the case in WSNs, a node may be completely disconnected from the network by removing a small number of edges, and as a consequence may have very few edge-disjoint paths.

4.3.2 Synthetic Network Illustration

We first illustrate our multi-path routing algorithms on randomly generated synthetic networks. In these networks nodes were distributed uniformly at random in a rectangular area, and an edge defined between each node and its three nearest neighbours. This process results in networks in which every node is attached to at least three others, however as the nearest neighbour relation is not symmetric, some nodes may be connected to more than three neighbours. Transmission and reception costs associated with each link were selected randomly from 5 configurations. The most energy required for a single transmission amounted to approximately 2.77 times the quiescent current expended between transmissions, while the least energy was 0.17 times the quiescent energy. Note that while nodes on the periphery of the network will make only a single transmission per reporting cycle, those closer to the base station may make many transmissions and receptions as they relay other nodes' data.

In the illustration presented here the network comprised 11 nodes plus a base station and we allow two paths per node, making it a 2-path routing scheme. Figure 4.5(*left panel*) shows the connectivity map for this network. For simplicity, all node batteries had the same initial charge and all edges were assumed to have the same link failure probability. In subspace pruning, we used $k = 10$ shortest paths for building each $\mathcal{P}_K$ and $\mathcal{P}'_K$; in addition we used braided and edge-disjoint path pruning to build the complete path library $\mathcal{P}$. The initial evolutionary population was created by randomly selecting 100 routes $\mathcal{R}$ from $\mathcal{P}$. Then solving the linear programs (4.7a) and (4.14a) generated distinct time shares $\mathcal{T}_L(\mathcal{R})$ and $\mathcal{T}_F(\mathcal{R})$ respectively for each route, resulting in 200 initial solutions $(\mathcal{R}, \mathcal{T})$. The initial archive comprised the non-dominated solutions among these random solutions. During the evolutionary optimisation we used $c = \mu = 0.1$ for the crossover and perturbation rates. After 40000 generations, the Pareto front estimation was judged to have been well-converged based on the dominated hypervolume measure [Zitzler, 1999].

In Figure 4.5, the estimated Pareto front from the evolutionary optimisation is shown. As the figure shows, the 2-path routing schemes in the estimated Pareto set not only contains a wide range of solutions representing various levels of trade-off between network lifetime and robustness, but also performs better than solutions in the random initial archive. Note that the initial random archive was constructed from solutions within the *pruned* subspace and by solving the linear programs in 4.7 and 4.14. Routing schemes selected at random from the entire search space with all possible routes and arbitrary time shares between the constituent routes, perform substantially worse in comparison (see Figure 4.5). With the estimated Pareto optimal solutions, the network engineer may inspect the solutions

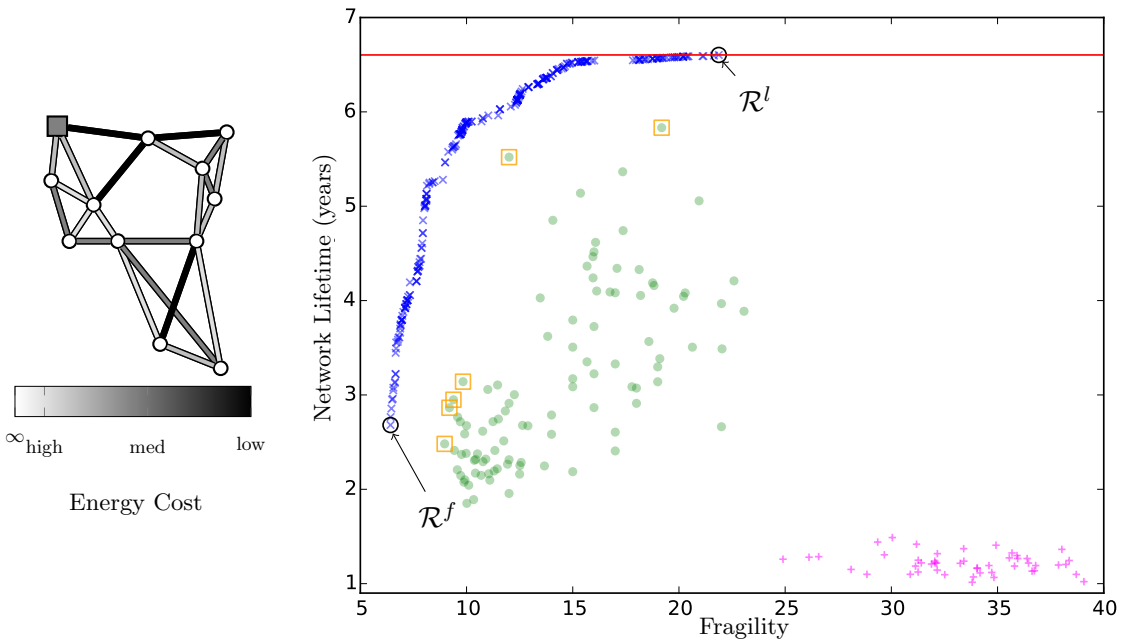


Figure 4.5 Pareto front approximation showing the trade off between network lifetime and fragility, resulting from the evolutionary multi-path routing optimisation in a random network with 2 paths per node, for the network on the left. The Pareto front approximation is shown with blue crosses, with $\mathcal{R}^l$ and $\mathcal{R}^f$ being the best network lifetime and the most robust solutions respectively. Initial random solutions are depicted with solid green dots and the non-dominated solutions in the initial archive are indicated with empty orange square around the associated random solutions. The magenta pluses depict solutions selected at random from the entire search space with the time shares between routes assigned arbitrarily, and they perform significantly worse in comparison to the random archive used here. The red line shows the upper bound for network lifetime if unlimited paths per node are permitted.

and select a suitable multi-path routing scheme for this particular network.

The best lifetime solution $\mathcal{R}^l$ achieves a network lifetime of 6.6 years with two routes allowed per node; this is within 0.001% of the theoretical upper bound of the network lifetime for this particular network, which can be calculated when there is no limit to the number of paths available to each node. At the other extreme, the evolutionary algorithm has located a comparatively robust solution reducing the fragility to 29.3% of best lifetime solution, at the expense of a 59.4% decrease in network lifetime. In most circumstances the network operator will want to choose a solution between these two extremes. Inspection of the figure shows that a solution close to the "knee" in the Pareto front (fragility $\approx$ 9.9, lifetime $\approx$ 5.9) may be preferred because decreasing the fragility further leads to a rapid deterioration in lifetime, whereas large increases in fragility are required to achieve significantly longer lifetimes.

To clarify the nature of both objectives and their relationship with multi-path routing schemes, we further visualise in Figure 4.6 the routes corresponding to the solutions $\mathcal{R}^l$ and $\mathcal{R}^f$, which optimise the lifetime and fragility respectively. This figure shows the connectivity map for the network (Figure 4.6a), the overall edge utilisations for $\mathcal{R}^l$ and $\mathcal{R}^f$ (panels 4.6b and 4.6c) and each node's paths in $\mathcal{R}^l$ and $\mathcal{R}^f$. Time shares τ_{i1} and τ_{i2} allocated to each of the two paths from a given node v_i are indicated using the colour scale, green for $\mathcal{R}^l$ and red for $\mathcal{R}^f$.

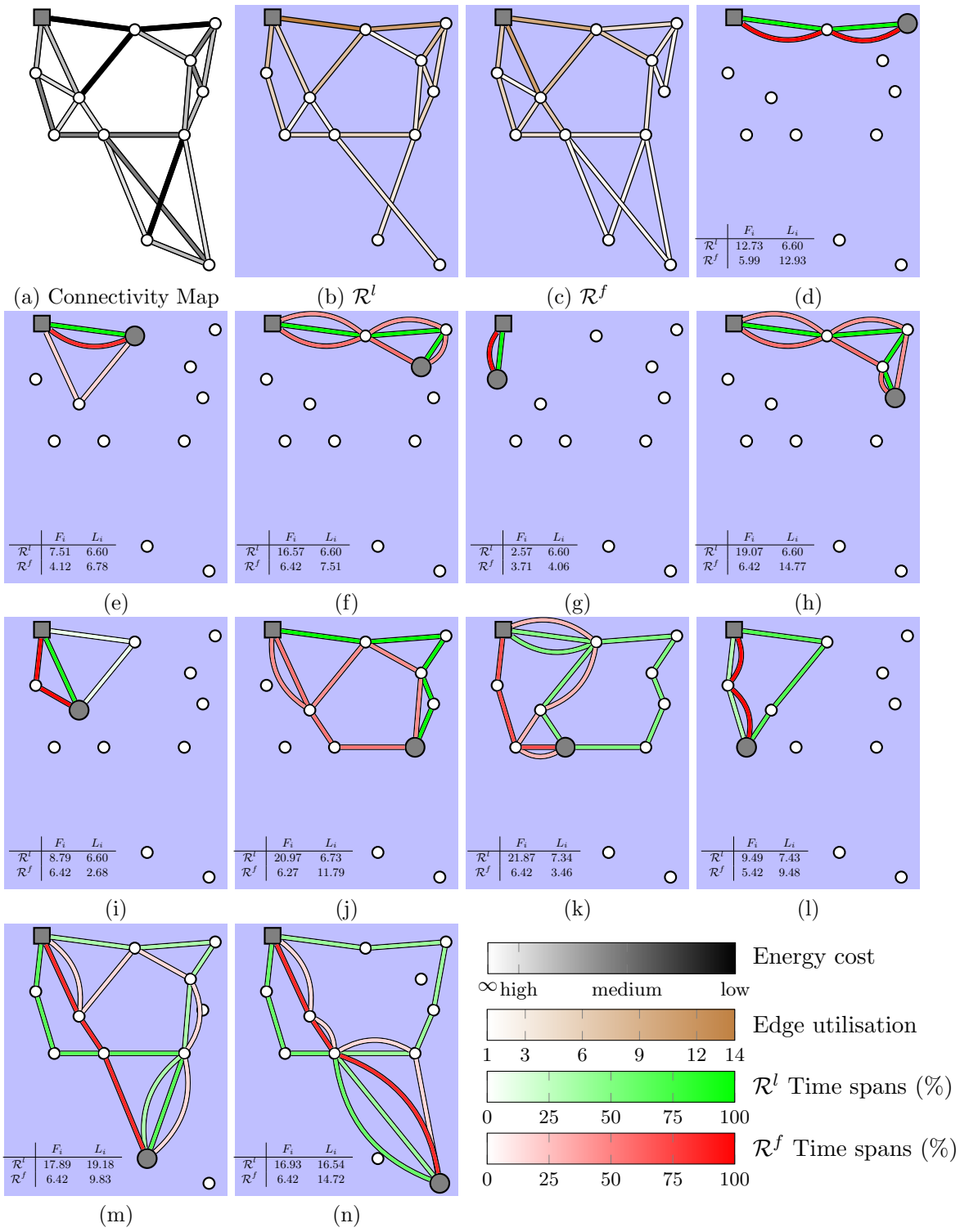


Figure 4.6 Comparison between extreme solutions: best lifetime solution ($\mathcal{R}^l$) and most robust solution ($\mathcal{R}^f$) in the median run of 31 runs for the random network; see Figure 4.5. (a) shows the connectivity map with all available links between nodes and associated energy costs (darker represents lower energy cost), with the grey square node indicating the base station. (b) and (c) depict the edge utilisations for all routes in solutions $\mathcal{R}^l$ and $\mathcal{R}^f$ respectively (darker represents higher utilisation). Each of (d) – (n) shows the active time shares between a pair of paths forming the solutions $\mathcal{R}^l$ (shades of green) and $\mathcal{R}^f$ (shades of red) for individual nodes (solid grey), with the associated lifetimes and fragility written at the bottom left. Only paths with non-zero time shares are shown. Overall $\mathcal{R}^l$ prefers energy efficient single routes with only few nodes using multi-paths to distribute load from most heavily loaded nodes, while $\mathcal{R}^f$ mainly balances traffic to form disjoint paths and improve fragility irrespective of energy costs.

The edge utilisations for $\mathcal{R}^l$ in Figure 4.6b indicate that energy efficient edges are utilised to achieve long network lifetime. This is clear when edge utilisations are compared with the connectivity map in Figure 4.6a: higher edge utilisations occur where energy costs are relatively low. This is also evident from inspecting individual node's paths: for 6 of the 11 nodes $\mathcal{R}^l$ uses single-paths, i.e. only a single path is used to send all the data ($\tau_{i1} = 1$ and $\tau_{i2} = 0$). Multiple paths tend only to be used for nodes distant from the base station (Figures 4.6k-4.6n) in order to relieve more heavily loaded nodes closer to the base station.

By contrast, edge utilisations for the least fragile routing $\mathcal{R}^f$ (Figure 4.6c) show no obvious relationship with energy costs, although nodes close to the base station are inevitably used more heavily than distant nodes. Instead, it is clear that robustness is achieved by providing two paths for 7 of the 11 nodes. Where only a single path is used for a node (e.g., Figure 4.6g), we have verified that adding an additional path increases the *overall* fragility of the network because it requires additional traffic to use an already vulnerable link. It is, of course, possible to ensure that all nodes incorporate some redundancy by using distinct paths at least a fraction τ_{min} of the time by replacing the positivity constraints (4.7c) and (4.14c) with $\tau_{kd} \geq \tau_{min}$ for all k, d.

In the left hand panel of Figure 4.7 we present the summary attainment surfaces (see section 2.3.4, Fonseca and Fleming [1996], and Knowles [2005]). These are the Pareto fronts achieved by 10%, 50% and 90% of 31 independent runs of the evolutionary optimiser. Clearly, the attainment surfaces for 2-path schemes are superior to those from single-path schemes in terms of both lifetime and fragility. We attribute this to the possibility of load balancing between the paths for a single nodes as well as between nodes. Also, the narrow width of the attainment surfaces indicates desirable repeatability and convergence properties of the proposed evolutionary approach.

4.3.3 Real Network Illustration

Finally, we show the performance of the approach on the real network deployed in the Victoria & Albert Museum, London as discussed in chapter 3.[3] The network comprises 30 sensor nodes and a base station. Nodes are connected to between 3 and 21 other nodes, with the average degree being 11.9. The characteristics of the radio environment also vary with the passage of visitors through the galleries (over 3 million visitors passed through the museum in 2014), which may lead to occasional link failures.

We used the same evolutionary algorithm configuration as described for the synthetic network for evolutionary multi-path routing optimisation in this network; except 60000 function evaluations were required to achieve similar convergence quality as indicated by the dominated hypervolume measure [Zitzler, 1999]. Routing optimisation, using two paths per node, resulted in a range of trade-off solutions, with the best lifetime solution in the median run of 31 optimisations achieving 99.51% of the theoretical best network lifetime with unlimited paths. In this network the quiescent energy expenditure is higher,

[3] Data on the network to allow comparison with this work can be found at `http://emps.exeter.ac.uk/computer-science/wsn/`

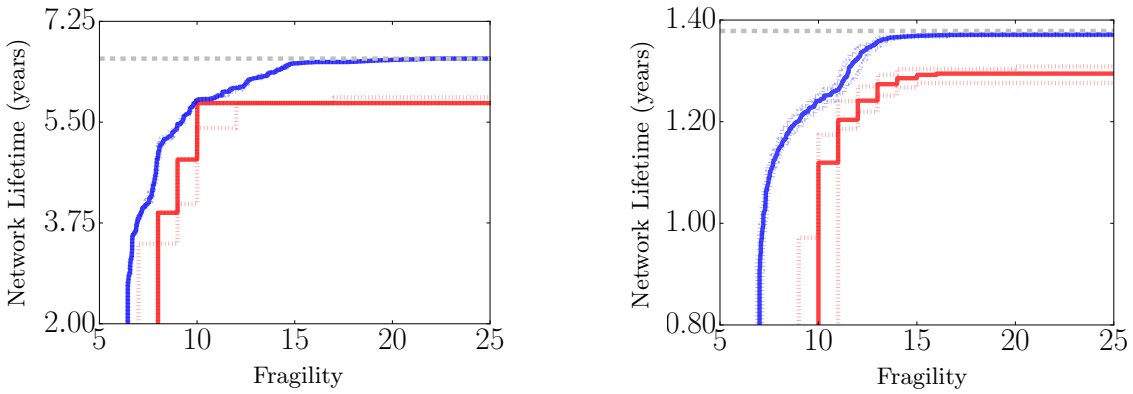


Figure 4.7 Performance comparison between single-path routing schemes (red) and 2-path routing schemes (blue) approaches. *Left*: Random network, *Right*: Victoria & Albert Museum network. The solid lines depict 50% summary attainment surfaces and dotted lines show 10% and 90% surfaces. The upper bound of the maximum minimum lifetime for the network is shown with horizontal dashed line in light grey. In both cases the multi-path schemes completely dominates the single-path schemes.

so that transmission and reception energy costs amount to between 0.017 and 0.27 times the quiescent energy expended in a reporting cycle. Consequently, the maximum available lifetime and the range of available lifetimes is smaller than in the synthetic network where transmission and reception costs are more significant. Nonetheless, the evolutionary search has located a wide range of lifetimes and shows the trade-off with robustness. At the other extreme, the fragility of most robust solution was 37.9% of the robustness of the best lifetime routing at an expense of a reduction of 29.5% in network lifetime in comparison with the best lifetime solution.

The right hand panel of Figure 4.7 shows the $10\%, 50\%$ and 90% summary attainment surfaces from 31 optimisation runs for one and two paths per node routings. As for the synthetic network the close proximity of the attainment surfaces, indicates the repeatability and convergence of the optimisation. The optimised 2-path routings clearly dominate the single path routings, providing better lifetimes and robustness.

Note that, in this network, allowing an additional path for each node does not provide significantly better routings (see Figure 4.8), and in fact many time share distributions in 3-path routings indicate only 2 paths are being utilised. Furthermore, the best network lifetime solution with unconstrained 2-path scheme in the median run achieves 99.51% of the theoretical best lifetime.

Also, recall that in section 3.6.3, a constrained version of 3-path scheme where nodes were only allowed to change routes together, achieved a best lifetime equal to 99.2% of the theoretical best. The unconstrained 2-path scheme can achieve similar performance without the use of an extra route per node. We attribute this to the greater flexibility of unconstrained multi-path schemes where nodes may change routes without depending on other nodes.

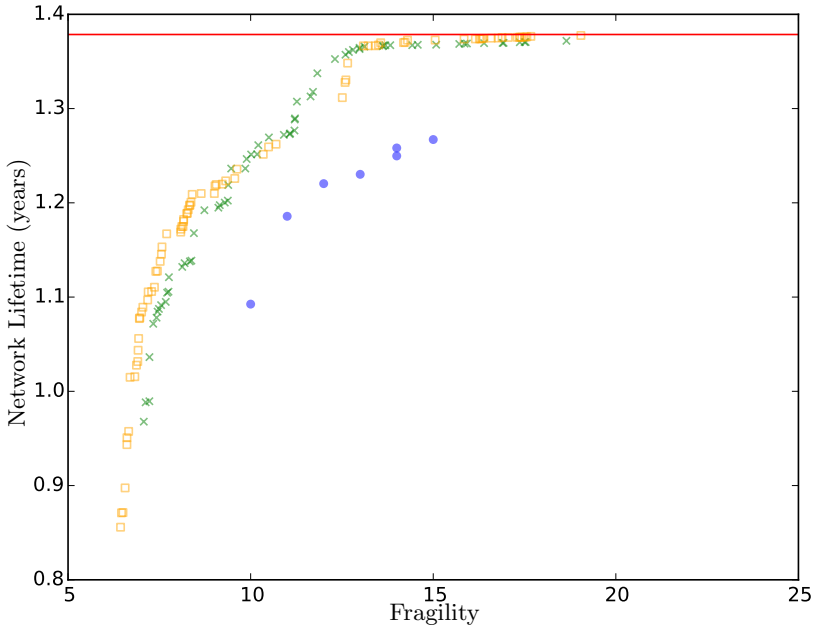


Figure 4.8 Pareto front approximation comparison between 1-path (blue dots), 2-path (green crosses), and 3-path (orange empty squares) routing schemes for a single run. While using more than one path significantly improves performance; allowing additional paths over 2-path routing scheme does not substantially improve performance.

4.3.4 Local Lifetime and Fragility Optimisation

Finding the time shares best for either network lifetime or robustness amounts to solving the linear programs (4.7) and (4.14) for the time shares $\mathcal{T}$. While obtaining a solution is straightforward and rapid, it turns out that for many networks not all the time shares for all the nodes are uniquely determined by the solution, that is there may be multiple optimal solutions to this problem.

To see how this may occur consider a network in which the minimum lifetime node is v_i and there is another node v_j, but none of the routes passing through v_j also pass through v_i; this might be because v_i and v_j are on geographically opposite sides of the network, for example. In this case changing time shares for v_j's routes has no impact on the lifetime of v_i, and v_i remains the optimal minimum lifetime node. Consequently, the time shares for v_j are not determined by the solution to (4.7), although a numerical solver (e.g. Gurobi [2015] or CPLEX [2015]) will generally set the relevant variables to arbitrary values.

Similarly, when considering the fragility, if the routes from two nodes v_i and v_j have no edges in common and v_i is the node found to have the maximum fragility by the solution to (4.14), then the time shares for v_j have no bearing on fragility of the network as a whole and are undetermined by the solution to the linear program. In each of these cases, although the network-wide lifetime or the fragility cannot be improved, there is freedom to choose the undetermined (slack) time shares $\tau_{id} \in \mathcal{T} \setminus \{\tau_{i'd'}\}$ to improve the *local* performance, where $\tau_{i'd'} \in \mathcal{T}$ are the time shares determined by solving corresponding LP.

One way to identify the time shares τ_{id} and $\tau_{i'd'}$ is to consider the total derivatives of the fragility in (4.14) or the inverse lifetime in (4.7) with respect to the time shares; this

works because the time shares with no impact on network-wide fragility or lifetime will result in a total derivative of zero. However, the approach fails for problems involving more than 2-paths: in this case the problem becomes under-constrained, and thus the dependent variables for total derivative can not be readily replaced (see equations (4.7d) and (4.14d)). We therefore utilise vertex enumeration for this purpose.

The space of feasible solutions to a linear program is well known to be a convex polyhedron [Bertsimas and Tsitsiklis, 1997; Boyd and Vandenberghe, 2004]. The solution is either a vertex of the polyhedron (if it is unique and all the τ_{id} are determined) or a convex polyhedral facet on which the objective takes the optimum value [Miller, 1963; Bertsimas and Tsitsiklis, 1997]. Numerical solvers return an arbitrary vertex on this polyhedral facet. However, the number of τ_{id} which are determined at particular vertices of the optimum polyhedral facet will vary according to the vertex because different constraints ((4.7d) and (4.14d)) are active. We therefore explore the optimal facet's vertices using a vertex enumeration algorithm (e.g. Avis and Fukuda [1992]) to find the solution which determines the smallest number of active τ_{id}. Note that vertex enumeration algorithms commonly use a similar tableau format to the simplex solvers for linear programs, so that the latter may be readily adapted for vertex enumeration beginning at one of the vertices of the optimum polyhedral facet. The vertex enumeration approach does not suffer from the limitations of the total derivative approach, and may be adopted as a generic approach. Once the active variables are identified, the linear program may be solved again but with the active τ_{id} constrained to their determined values.

Solving the constrained LP will fix additional τ_{id}, but may not determine all of them if they are loosely coupled, as is usually the case. In this case, a deflationary procedure may be deployed to determine all time shares. The process starts by solving the associated LP ((4.14) or (4.7)) to find a vertex on the optimal polyhedral facet. The vertex enumeration process is then applied to locate all vertices on the facet, and thus the vertex with the minimum number of active constraints is located. This vertex may then be considered as the solution that determines the least number of time shares. The LP is then constrained with the fixed time shares from this solution, and solved again with the freedom to vary only the undetermined time shares. As this may not determine all of the remaining time shares, the process is repeated until all time shares are determined. This process is summarised in Algorithm 4.2.

We note that the new vertices found through vertex enumeration have the same optimal objective value for the associated LP. If the subspace (of the decision space), on which the other objective is constant, is parallel to the part of the optimal polyhedral facet bounded by the contiguous vertices, then traversing between such vertices results in the same objective value for the other objective. In other words, the potential changes in the time shares due to traversing between these vertices have no impact on the objective value for both objectives. Although these vertices are distinct in the decision space, they have the same objective value in both objectives, and therefore overlap in the objective space. In contrast, when the subspace, on which the other objective is constant, is not parallel to the part of the optimal polyhedral facet, the other objective value changes from one vertex to another. As a result a vertex is either better or worse in comparison to its immediate

Algorithm 4.2 Linear program deflationary procedure. The function ConstrainedLPSolve($\mathcal{T}, \tilde{\mathcal{T}}$) denotes a numerical LP solver for problem (4.7) or (4.14), with variables $\tilde{\mathcal{T}}$ fixed.

Inputs

$\mathcal{T}$: Set of time shares to be determined

1: $\tilde{\mathcal{T}} = \emptyset$ ▷ Determined $\mathcal{T}$
2: **while** $| \mathcal{T} \setminus \tilde{\mathcal{T}} | > 0$ **do**
3: $\boldsymbol{\tau} \leftarrow$ ConstrainedLPSolve($\mathcal{T}, \tilde{\mathcal{T}}$)
4: $V \leftarrow$ VertexEnumerate($\boldsymbol{\tau}$)
5: $\boldsymbol{\tau} \leftarrow \text{argmin}_{\boldsymbol{\tau}' \in V}$ NumberActiveConstraints($\boldsymbol{\tau}'$)
6: $\tilde{\mathcal{T}} \leftarrow \tilde{\mathcal{T}} \cup \{\boldsymbol{\tau}\}$
7: **end while**
8: **return** $\tilde{\mathcal{T}}$ ▷ Fully determined time shares

neighbours in terms of the other objective, while still maintaining the optimal objective value for the associated LP. These effects of vertex enumeration on the other objective are illustrated in Figure 4.9. With this discussion, it can be envisaged that the initial vertex may be suboptimal in terms of the other objective. Therefore, while selecting the vertex with the least number of active constraints (step 5 in Algorithm 4.2), it is necessary to consider only the vertices that improve, or at least maintain, the other objective.

We also note that the number of vertices per polyhedral facet varies from one to thousands, although the exact number of vertices that must be explored is initially unknown. Therefore, even running the full deflationary procedure for a routing scheme may prove to be expensive, especially if there are many vertices to explore. Since the LP solver may produce a suboptimal initial vertex in the other objective, it is still necessary to enumerate the vertices for the routing schemes in the final estimated Pareto front as it would aid in discarding any suboptimal vertices in the other objective.

In Figure 4.10, we use only vertex enumeration for the solutions in a final estimation of the Pareto front for the network discussed in section 4.3.2 with 2-path routing schemes. The results show that all solutions generated by only the evolutionary procedure are also optimal vertices in terms of the other objective. We attribute this to the evolutionary pressure, that is the pressure exerted by the evolutionary mechanism to select routing schemes for which the LP solver produces the optimal vertex in the other objective. Therefore, the additional check on the routing schemes from the final estimation of the Pareto front to ensure that none of the initial vertices are suboptimal in the other objective by using vertex enumeration is not necessary.

An alternative approach to improve local performances is to use evolutionary algorithms to determine the undetermined time shares whilst constraining the search within the optimal polyhedral facet. This approach, however, is likely to be suboptimal, and it is expected that the deflationary procedure with vertex enumeration would be efficient if there are few vertices to explore.

Future work might involve investigating the full deflationary process in order to explore the extent of local performance improvement that may be achieved. However, we emphasise

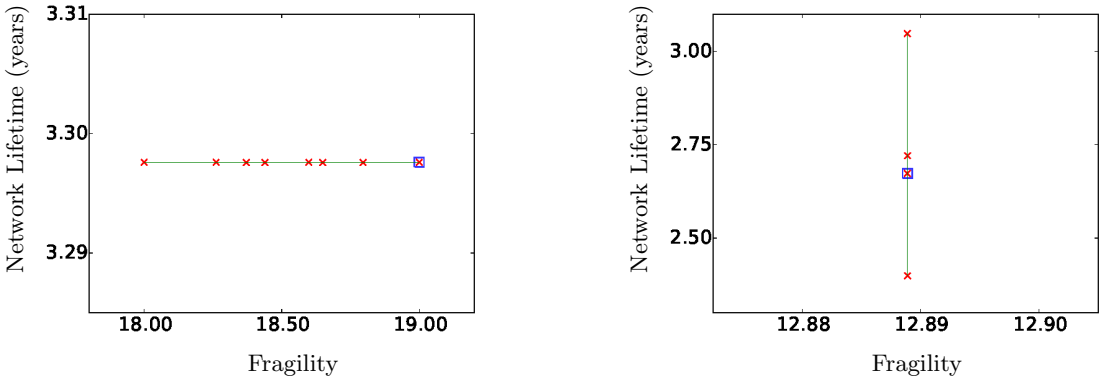


Figure 4.9 Nature of vertex enumeration for two 2-path routing schemes $\mathcal{R}^l$ and $\mathcal{R}^f$ from random initial archive for the network discussed in section 4.3.2. *Left:* all (832) vertices for $\mathcal{R}^l$ with time shares solved for lifetime linear program; *right:* all (96) vertices for $\mathcal{R}^f$ with time shares solved for robustness linear program. The initial vertex from appropriate solver is shown with empty blue square, new vertices found through enumeration are depicted with red crosses, and the green line connecting all vertices show that they belong to the same polyhedral facet. The initial optimal objective value for the associated LP does not change while conducting vertex enumeration (perpendicular to corresponding axis). However, the opposing objective value may differ among vertices with the initial vertex potentially being suboptimal in this regard.

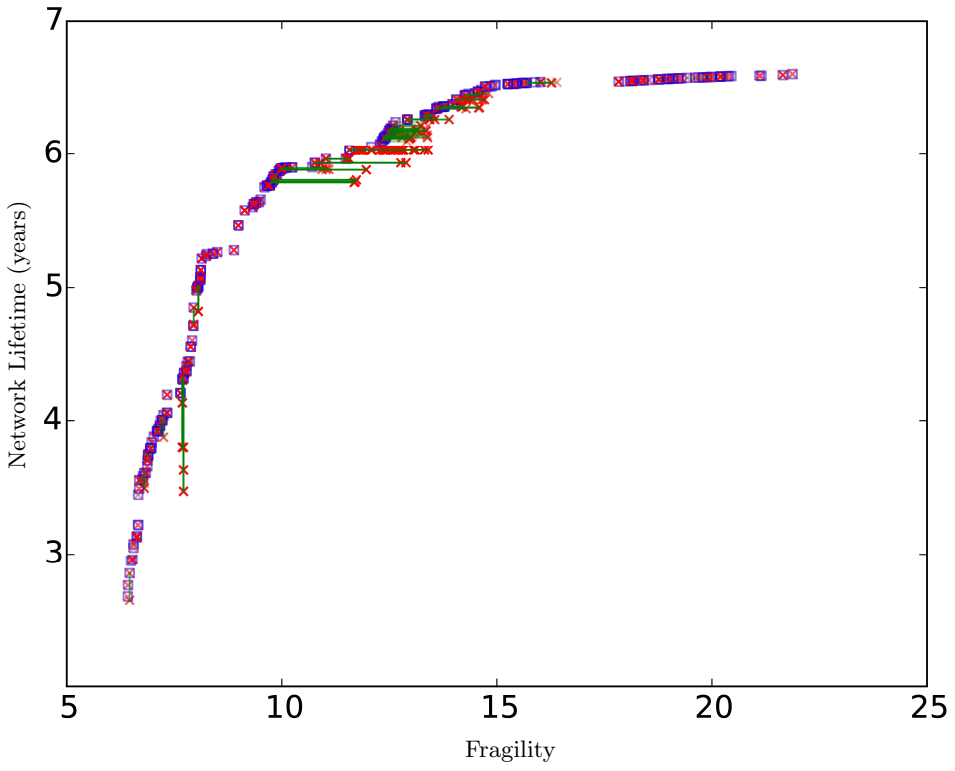


Figure 4.10 Illustration of vertex enumeration for the synthetic network presented in section 4.3.2 with 2-path routing schemes. Solutions are included from the final estimation of Pareto front. The blue squares show the initial vertex from solver, the red crosses portray all discovered vertices, and the green line connects all vertices within the same polyhedral facet. Due to the evolutionary pressure, none of the initial vertices from the estimated Pareto front are suboptimal in the other objective.

that this procedure need only be undertaken after a routing scheme has been selected by the evolutionary algorithm as the network-wide quality (network lifetime and fragility) of the solutions may remain unaffected, and as the results suggest there is very little to gain from using this method with potentially great computation cost.

4.3.5 Bi-Objective Linear Program (BOLP)

The evolutionary approach discussed so far is capable of generating a wide range of solutions representing varying degrees of trade-off. Recall that during the evolutionary process, a set of routes is proposed through the evolutionary mechanism, and then the LPs in (4.7) and (4.14) are solved independently to generate associated sets of optimal time shares. This process effectively generates two (extremal) solutions, each of which is then compared with the current members of the archive, and non-dominated solutions are retained (step 6 in algorithm 4.1). These two linear programs may be treated as a *Bi-Objective Linear Program* (BOLP) instead, and thus there is an optimal Pareto front with potentially many solutions lying between these two extremal solutions. It may be desirable to augment the estimated Pareto front from the evolutionary approach with the intermediate solutions from the BOLP, and it is possible that some of the BOLP front solutions may dominate the estimated Pareto front obtained using only the extremal solutions. We investigate this idea in this section.

The simplest approach to locate the Pareto front of this BOLP is to reduce it to a straightforward weighted sum of the two objectives, and generate a *Single Objective Linear Program* (SOLP), where the weights associated with each objective represents the relative importance of the objectives [Boyd and Vandenberghe, 2004; Ehrgott, 2006]. Combining the LPs in equations (4.7) and (4.14), the SOLP may be expressed as:

$$\min_{\mathcal{T}} \left(\lambda L^{'*} + (1-\lambda) F^{*} \right) \tag{4.18a}$$

subject to:

$$q_k L^{'*} - N_c \left(B_k + \sum_{R_{id} \in R} C_{id}^{k} \tau_{id} \right) \geq 0 \quad \forall k; \tag{4.18b}$$

$$F^{*} \geq F_{id} \quad \forall R_{id} \in \mathcal{R}; \tag{4.18c}$$

$$\tau_{kd} \geq 0 \quad \forall k, d; \tag{4.18d}$$

$$\sum_{d} \tau_{kd} = 1 \quad \forall k; \tag{4.18e}$$

where $0 \leq \lambda \leq 1$ is the weight that controls the relative importance of the inverse lifetime $L^{'*}$ and the fragility F^{*}.

Solving this SOLP results in a single point on the optimal front. Clearly, when $\lambda = 0$, it is effectively the same as solving the optimal time share problem for robustness. Similarly, when $\lambda = 1$, it is equivalent to solving for optimal time shares that maximise the network lifetime. The optimal trade-off front may be discovered from one end of the front to other by varying λ. In Figure 4.11, the result of applying weighted sum on all the solutions from the estimated Pareto front for the synthetic network is shown. Clearly, the newly generated solutions interpolate between solutions in the estimated front. In this case the BOLP augmented front does not lie significantly ahead of the base front, although larger gains might be achieved for different networks.

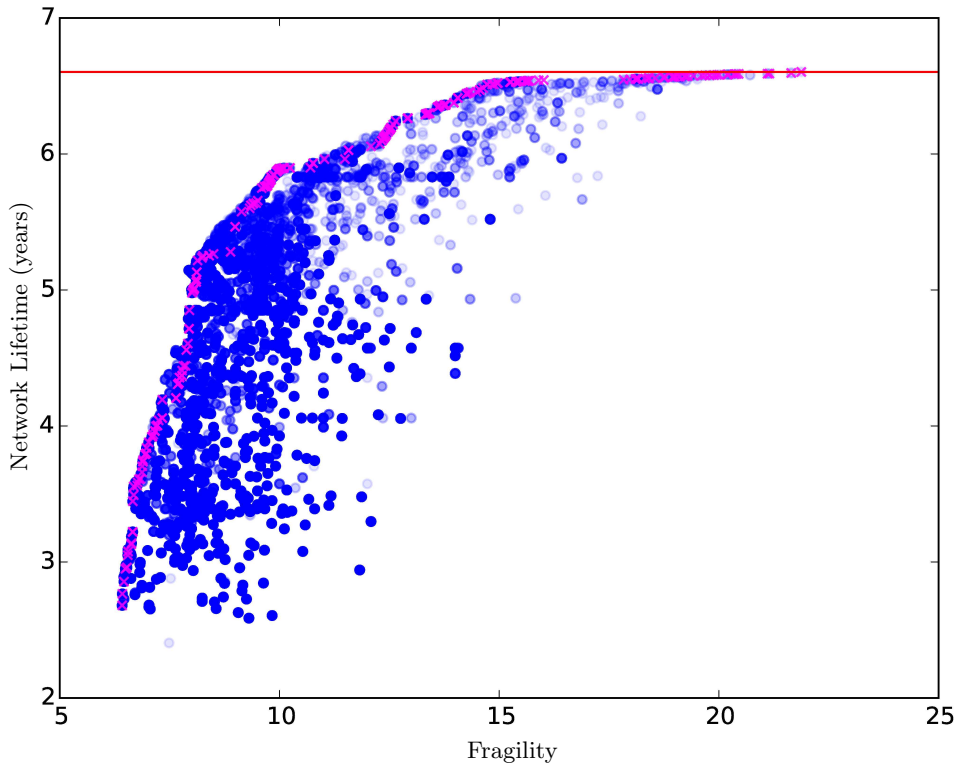


Figure 4.11 Effects of applying weighted sum on the solutions from the estimated Pareto front (magenta crosses) for the synthetic network discussed in section 4.3.2 with a resolution of 0.001 in varying λ. All solutions (blue dots) generated through weighted sum interpolates between solutions from the estimated Pareto front. Incorporating these solutions does not improve Pareto front approximation significantly.

The optimal Pareto front of the BOLP is a segment of a convex polyhedron in two-dimensional objective space, and to discover the complete front it is sufficient to locate the vertices of this facet [Ehrgott, 2006]. The parametric bi-objective simplex algorithm is capable of identifying such vertices [Ehrgott, 2006]. Rather than incorporating many solutions from the front by varying λ, just considering the vertices of the front may be a more sensible approach in order to potentially improve the Pareto front approximation. However, the computational overhead due to such additions to the overall evolutionary approach is likely to be very expensive, and as this makes little practical difference, we do not investigate this further.

4.4 Summary

Despite advances in battery technology it remains important to extend the lifetimes of wireless sensor networks by choosing routes that balance the loads placed on individual nodes. However, as we have shown here, optimisation solely of the lifetime may be detrimental to the robustness of the network to link failure. In this chapter we have therefore presented novel methods for discovering the best trade off between network lifetime and network robustness with unconstrained multi-path routing schemes.

Chang and Tassiulas [2004] elegantly showed how the network lifetime may be maximised by solving a linear program. However, their work is applicable only when each node may use an unlimited number of paths, which may be impractical for low-power, limited memory devices. In contrast, limiting the number of paths results in an NP-hard combinatorial problem. Nonetheless, in the example networks presented here, our multi-objective evolutionary algorithm is able to locate routes with network lifetimes within 1% of the Chang *et al.* lifetime, even with only two paths per node. Undoubtedly there will be networks that require more paths to approach the Chang *et al.* lifetime, but our experience suggests that two paths allows routes with very good lifetimes to be found.

Network robustness to link failure is quantified by the network *fragility*, which measures the maximum expected data loss if a link fails. This measure incorporates the probabilities of individual links failing. Initially, in the absence of any prior information these probabilities may be set equal, but during network operation links may be monitored to better estimate these probabilities. With these on hand, the optimisation may be repeated to minimise the fragility. In the case of protracted link failure, we have shown elsewhere [Rahat et al., 2014] that re-optimisation of the routing using the present solution as a base can cope with dynamic conditions.

Inspecting the robustness measure, i.e. network fragility, it is clear that a many-objective problem involving one objective per node with a view to improving node-wise fragility can be derived. In solving many-objective problems however, the major difficulties are: the search ability deteriorates, and the number of solutions necessary to approximate Pareto front increases exponentially [Ishibuchi et al., 2008]. Therefore, we only considered one network-wide objective regarding robustness, that is to minimise the network fragility across all nodes. As the results in this thesis suggest, the network fragility measure in its current form can be a useful tool to analyse robustness for any network.

Evolutionary algorithms are an effective way to obtain good solutions to combinatorial optimisation problems. Efficient optimisation for this problem was obtained by two measures. First, the space of possible solutions was pruned to limit the search to routes likely to have good lifetime and/or robustness. Secondly, we showed how to find the optimal division of traffic between the paths from each node by solving a linear program for either lifetime or fragility. The resulting hybrid algorithm is efficient because the need to use a further, relatively slow, evolutionary process to find the optimal division of traffic is obviated.

Solving the optimal time share LPs for a given multi-path scheme may result in finding an optimal solution for respective LP with many alternative optima, and some time shares may be undetermined. Such optimal solutions lie on a polyhedral facet, and any of the vertices of the facet may be selected as the optimal solution. Moving from one optima to another provides an opportunity to minimise the number of active constraints, and thus increasing the number of undetermined time shares; this problem may then be reformulated as another LP with only the undetermined time shares treated as variables. Therefore, we may achieve better local lifetime or fragility as all time shares are determined through vertex enumeration and iterative deflationary LP solving process. This process may also

be undertaken to check whether the solutions in the final approximation of the Pareto front contain any suboptimal vertices with respect to the other objective. However, it is not desirable to incorporate this as part of fitness evaluation in the overall evolutionary process as this may prove to be exorbitantly expensive with very little practical gain.

Furthermore, in the evolutionary process, we only consider two extreme solutions by solving the optimal time share problems for lifetime and fragility separately. Treating these two LPs as a BOLP indicates that there is an optimal trade-off front with many potential solutions, which we do not consider and thus there are areas that are never explored by the evolutionary approach. As a consequence, some gaps may occur in the overall estimated Pareto front. Additionally, augmenting the estimated Pareto front with the solutions from the BOLP may improve the quality of the approximation. As Ehrgott [2006] suggested, the Pareto front of a BOLP is a convex polyhedral facet, and the vertices of this facet may be identified using parametric bi-objective simplex algorithm. As a potential extension, we propose incorporating all such vertices within our evolutionary method to reduce the discontinuities that may appear in the estimated Pareto front. This is however computationally expensive procedure that appears to make relatively little practical difference.

Chapter 5

Conclusion

With the advancement in micro-electro-mechanical embedded systems and battery technologies, wireless sensors are becoming more popular for industrial and regulatory monitoring applications, mainly because of the ease of installation and the capability to monitor locations that are difficult to access.

Modern deployments of wireless sensor networks increasingly use mesh network topologies, where sensors send data to the base station either directly or indirectly by relaying data through other nodes. Mesh networks not only extend the network range as nodes do not need to have a direct link to the base station so long their neighbours are connected, but also provide an avenue to deal with the dynamic radio environment through alternative routes in case of node or link failure.

In mesh networks, if paths to the base station use expensive links – links with higher power and lower baud rate configurations – then the total energy consumption in the network will be high. Therefore, networks are often designed with a view to reduce overall energy consumption throughout the network; this is equivalent to maximising average lifetime of nodes.

Using only energy efficient routes may require directing most traffic through a few nodes, especially the nodes that are close to the base station. As a consequence, these nodes will exhaust their batteries quickly, and manual intervention would become necessary to replace batteries. It is therefore important to extend the time before any node exhausts its battery – the network lifetime – to enable load balancing and infrequent maintenance activities. Chang and Tassiulas [2004] formulated an LP for maximising the network lifetime when there is no limit on the number of paths that may be used by a node. However, for practical centralised systems this may not be achieved due to limited computational and memory resources at sensor nodes.

Clearly, extending network lifetime is in conflict with energy efficiency objective, and thus there is a trade-off to consider while selecting routing schemes. This multi-objective problem of extending network lifetime and promoting energy efficiency simultaneously requires considering all possible routing schemes. This combinatorial problem is NP-hard.

In this thesis, we therefore presented a hybrid evolutionary approach to estimate the optimal trade-off front. To combat the potential combinatorial explosion, we limited the search space using various pruning methods, namely k-shortest path pruning to retain energy efficient paths, and graph reduction based on the LP formulation for network lifetime to retain paths with long lifetimes.

Most routing optimisation approaches consider single-path routing schemes: a set of routes where each node has a route to the base station. In this thesis, we showed that an evolutionary approach may be used to locate optimal routings with single-path routing schemes. However, single-path schemes are somewhat limited in balancing the loads. Hence we extended this to incorporate multi-path routing schemes, where each node has more than one route to the base station. In chapter 3, this multi-path scheme was constrained as all nodes changed routes together, effectively switching between multiple single-path routing schemes. We also showed that a linear program may be solved to calculate the optimal time shares between these constituent single-path schemes. Results showed that with only three paths allowed per node, we can approximate within 1% of the optimal network lifetime when there is no limit on the number of routes, and significantly better the approximation over single-path schemes in a network deployed in the Victoria & Albert Museum.

As wireless sensors are often placed in dynamic radio environments, link failures are unpredictable and data may be lost due to link failures as routes cease to function, even if multi-path routing schemes are used. In such instances, a straightforward re-optimisation of networks may be used to re-establish network operation with current connectivity information (see, for example, Rahat et al. [2014]). Nonetheless, using multi-path routing schemes provides an opportunity to split the traffic between different routes such that the expected data loss in the event of link failure – the fragility – is minimised. Minimising the fragility therefore improves network robustness against link failures. The fragility measure introduced in this thesis is associated with the routes. It is essentially the total data loss across all routes that may share the failed links, and calculated using link failure probabilities. This is designed to promote robust multi-paths that share as few links as possible so that some data may be retrieved in case of link failure.

Note that the link failure probabilities may not be available readily, especially when the network is being established, and it may be assumed that all edges are equally likely to fail. Nonetheless, during normal operation of the network, links that are being used by different routes may be monitored and associated failure probabilities may be calculated. The network may later be re-optimised with this empirical information to achieve a more robust routing scheme.

Designing robust networks requires paths to be as disjoint as possible without any regards to energy efficiency or network lifetime. Therefore, promoting robustness is in conflict with both energy efficiency and network lifetime objectives. As network lifetime is often deemed as more important than energy efficiency in practical implementations, we investigated the trade-off between network lifetime and network robustness.

Optimising for robustness and network lifetime is again an NP-hard multi-objective routing optimisation problem, and thus we used a hybrid evolutionary algorithm to estimate the optimal Pareto front. In addition to k-shortest path pruning and max-min lifetime pruning, we used braided and edge-disjoint paths to limit search space and retain robust paths in the search space. Note that although the energy efficiency objective is not directly addressed, using k-shortest path pruning implies that paths available for selection are fairly energy efficient. We also showed that an appropriate linear program may be solved for robustness or network lifetime once an unconstrained multi-path scheme is proposed by the evolutionary mechanism. Solving the linear programs makes the evolutionary approach more efficient as it is needless to conduct an additional evolutionary search to determine suitable time shares which would be significantly more time consuming.

Additionally, we discussed how local performance improvements may be achieved through vertex enumeration, and solving the bi-objective linear program combining robustness and network lifetime. Our approach was successful in locating a wide range of multi-path routing schemes with varying levels of robustness and network lifetime with the best network lifetime within 1% of the upper bound for network lifetime, and in this instance only with two routes allowed per node in the Victoria & Albert Museum network.

Although we illustrated our approach in a small wireless sensor network with 30 sensors, our formulations are not restricted to the number of nodes and can be easily used in its current form for larger networks. We also emphasise that our novel fragility measure can be used in other graph based problems with link uncertainty where route robustness or traffic load balancing through disjoint routes is important. For instance, a straightforward adaptation of the fragility measure may be used to minimise congestion in a network as using multi-paths may ease congestion. Furthermore, the techniques used in the evolutionary mechanism may be applied to other similar problems. The k-shortest path pruning method with appropriate edge cost parameters may be applied to any network based problems to limit search space. Also, the idea of conducting two-phases of evolutionary search; firstly in smaller local space, and, secondly in a combined wider space, in order to gain better convergence at the edges of the estimated Pareto front, may prove to be useful in other evolutionary search methods.

It should be noted that the full multi-path routing optimisation process may take several hours. Hence, initially the network may be configured with a routing scheme from non-dominated solutions generated during early stages of the optimisation process as this routing scheme should still be better than using any arbitrary routes. The optimisation may then continue, and at the end the network may be re-configured with an appropriate solution.

5.1 Future Work

In this thesis, we have demonstrated a hybrid evolutionary approach in wireless sensor networks that is capable of locating good routing schemes in terms of energy efficiency,

network lifetime, and robustness within a sensible time frame, making it viable for practical implementations. We discussed some possible future avenues in improving local performances in sections 4.3.4 and 4.3.5. Also improving hardware and radio technology to mitigate the challenges of WSNs, for example applying turbo coding for superior error correction performance [Berrou et al., 1993] or using new clock synchronisation methods to improve medium access performance [Lenzen et al., 2010], will always be beneficial. Here we briefly present the most relevant future research directions.

Non-linear Battery Model

In this thesis, we have considered a rather conservative linear battery discharge model. This is a good approximation and a standard practice in industry, even for lithium-ion batteries which show non-linear characteristics. Modelling non-linear discharge characteristics is a challenging problem, especially under intermittent discharge loads as prevalent in WSNs; see for example Castillo et al. [2004] and Yurur et al. [2015]. One way to tackle this modelling problem is to generate stochastic models under intermittent loads, see for instance Chiasserini and Rao [2001], or a regression model may be learnt from experimental data, e.g. Liu et al. [2012]. These models will introduce non-linearity in the network lifetime and energy efficiency formulations, and thus a linear program may not be solved for network lifetime. However, incorporating such models in the evolutionary approach may prove to be beneficial in terms of energy efficiency or network lifetime optimisation.

It should be noted that there is a significant drain at nodes due to quiescent operations. If this is reduced the true impact of optimisation may be observed; we demonstrated a wider range of solutions may be achieved when the quiescent consumption is lower in a synthetic network. Hence, using more energy efficient display devices, micro-controller and transceivers may yield greater benefits.

Distributed and Parallel Multi-Objective Optimisation

In this thesis, we have devised an approach for centralised systems, where the routing decisions are made at a central base station, and the routing information is distributed to the nodes. With improved batteries, memory resources and micro-controllers, sensors may afford to make routing decisions locally, and this can be then considered as a distributed multi-agent system, where agents (sensors) perform local actions in order to achieve a collective goal [Vinyals et al., 2011]. The nature of distributed algorithms indicates that a network may be flexible and self-configurable with changing environment and little external intervention, and therefore may be considered as an exciting avenue to explore.

Distributed multi-objective optimisation is a comparatively new field of research with few papers (see for example Clement et al. [2013]), and very little work has been done on WSNs [He et al., 2010]. In distributed multi-objective optimisation, transforming system-wide multiple objective into node specific multiple objectives is a challenging task, and He et al. [2010] presented a general decomposition method that may be used to achieve this

in WSNs. This technique may be used to decompose system-wide robustness and network lifetime to node specific objectives; however this may prove to be non-trivial.

To solve the resulting node specific multi-objective problem He et al. [2010] used standard scalarisation techniques, and their discussion is based on only one solution for a specific relative importance between objectives. This is because, without a decision maker available locally, the nodes are unable to choose an operating point, and communicating potentially large amounts of information regarding the Pareto set over the radio is infeasible in wireless sensor networks. Thus the trade-off information and the benefits of multi-objective optimisation are lost. Nonetheless, it may be interesting to investigate the distributed implementation of the optimisation problem discussed here.

A more interesting avenue to explore would be to use such decomposition methods in parallelising the multi-objective optimisation approach. Additionally, graph decomposition algorithms, e.g. [Raynal, 2013], may prove to be useful. Although parallel approaches to multi-objective optimisation is not new (see for example [Talbi et al., 2008]), but there is little work pertinent to wireless sensor networks.

With this discussion, an interesting extension to our work would be to develop decomposition based parallel multi-objective optimisation methods that can improve our current approach in estimating the optimal trade-off between robustness and network lifetime. At the very least, it is expected that the parallel approaches will speed the search in significantly large networks.

Bibliography

Adams, J. (2006). An Introduction to IEEE STD 802.15.4. In *Proceedings of the IEEE Aerospace Conference*, AERO '06.

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless Sensor Networks: A Survey. *Computer Networks*, 38:393–422.

Aljazzar, H. and Leue, S. (2011). K^*: A Heuristic Search Algorithm for Finding the k Shortest Paths. *Artificial Intelligence*, 175(18):2129 – 2154.

Avis, D. and Fukuda, K. (1992). A Pivoting Algorithm for Convex Hulls and Vertex Enumeration of Arrangements and Polyhedra. *Discrete & Computational Geometry*, 8(1):295–313.

Avis, D., Hertz, A., and Marcotte, O. (2005). *Graph Theory and Combinatorial Optimization*, volume 8. Springer Science & Business Media.

Baccour, N., Koubâa, A., Mottola, L., Zúñiga, M. A., Youssef, H., Boano, C. A., and Alves, M. (2012). Radio Link Quality Estimation in Wireless Sensor Networks: A Survey. *ACM Transactions on Sensor Networks*, 8(4):34:1–34:33.

Bader, J. and Zitzler, E. (2011). HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. *Evolutionary computation*, 19(1):45–76.

Banner, R. and Orda, A. (2007). Multipath Routing Algorithms for Congestion Minimization. *IEEE/ACM Transactions on Networking*, 15(2):413–424.

Bashyal, S. and Venayagamoorthy, G. K. (2007). Collaborative Routing Algorithm for Wireless Sensor Network Longevity. In *Proceedings of the Third International Conference on Intelligent Sensors, Sensor Networks and Information*, ISSNIP '07, pages 515–520.

Bellman, R. E. (1958). On a Routing Problem. *Quarterly of Applied Mathematics*, 16:87–90.

Berrou, C., Glavieux, A., and Thitimajshima, P. (1993). Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes. In *Proceedings of the IEEE International Conference on Communications*, volume 2 of *ICC '93*, pages 1064–1070.

Bertsimas, D. and Tsitsiklis, J. N. (1997). *Introduction to Linear Optimization*, volume 6. Athena Scientific Belmont, MA, USA.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA.

Bradstreet, L. (2011). *The Hypervolume Indicator for Multi-Objective Optimisation: Calculation and Use*. PhD thesis, University of Western Australia, Australia.

Brander, A. W. and Sinclair, M. C. (1995). A Comparative Study of k-Shortest Path Algorithms. In *Proceedings of the Eleventh UK Performance Engineering Workshop*, pages 370–379.

Bringmann, K. and Friedrich, T. (2008). Approximating the Volume of Unions and Intersections of High-Dimensional Geometric Objects. In *Algorithms and Computation*, pages 436–447. Springer.

Castillo, S., Samala, N. K., Manwaring, K., Izadi, B. A., and Radhakrishnan, D. (2004). Experimental Analysis of Batteries Under Continuous and Intermittent Operations. In *Proceedings of the International Conference on Embedded Systems and Applications*, ESA '04, pages 18–24.

Cerpa, A., Wong, J., Kuang, L., Potkonjak, M., and Estrin, D. (2005). Statistical Model of Lossy Links in Wireless Sensor Networks. In *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks*, IPSN '05, pages 81–88.

Chang, J.-H. and Tassiulas, L. (2004). Maximum Lifetime Routing in Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, 12(4):609–619.

Chehri, A., Fortier, P., and Tardif, P.-M. (2006). A Comparison Between Different FHSS Techniques for Use in A Multiple Access Secure Wireless Sensor Network. In *Proceedings of the IEEE Annual Wireless and Microwave Technology Conference*, WAMICON '06.

Chen, X., Makki, K., Yen, K., and Pissinou, N. (2009). Sensor Network Security: A Survey. *IEEE Communications Surveys and Tutorials*, 11(2):52–73.

Chiasserini, C. and Rao, R. (2001). Energy Efficient Battery Management. *IEEE Journal on Selected Areas in Communications*, 19(7):1235–1245.

Chien, C., Elgorriaga, I., and McConaghy, C. (2001). Low-Power Direct-Sequence Spread-Spectrum Modem Architecture for Distributed Wireless Sensor Networks. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 251–254.

Chong, C.-Y. and Kumar, S. (2003). Sensor Networks: Evolution, Opportunities, and Challenges. *Proceedings of the IEEE*, 91(8):1247–1256.

Cionca, V., Newe, T., and Dadarlat, V. (2008). TDMA Protocol Requirements for Wireless Sensor Networks. In *Proceedings of the Second International Conference on Sensor Technologies and Applications*, SENSORCOMM '08, pages 30–35.

Clement, M., Okimoto, T., Ribeiro, T., and Inoue, K. (2013). Model and Algorithm for Dynamic Multi-Objective Distributed Optimization. In Boella, G., Elkind, E., Savarimuthu, B., Dignum, F., and Purvis, M., editors, *PRIMA 2013: Principles and Practice of Multi-Agent Systems*, volume 8291 of *Lecture Notes in Computer Science*, pages 413–420. Springer Berlin Heidelberg.

Coello Coello, C. (1999). A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems*, 1(3):269–308.

Coello Coello, C. A., Lamont, G. B., and Veldhuizen, D. A. V. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. MIT Press.

CPLEX (2015). IBM ILOG CPLEX Optimizer. Available at `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html`.

Dantzig, G. B. (1963). *Linear Programming and Extensions*. Princeton university press.

Das, I. and Dennis, J. (1997). A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problems. *Structural optimization*, 14(1):63–69.

Deb, K. (2009). *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley paperback series. Wiley.

Deb, K. and Jain, H. (2014). An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Dijkstra, E. W. (1959). A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1(1):269–271.

Ehrgott, M. (2006). *Multicriteria Optimization*. Springer Science & Business Media.

Emmerich, M., Beume, N., and Naujoks, B. (2005). An EMO Algorithm Using the Hypervolume Measure as Selection Criterion. In Coello Coello, C., Hernández Aguirre, A., and Zitzler, E., editors, *Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 62–76. Springer Berlin Heidelberg.

Eppstein, D. (1999). Finding the K Shortest Paths. *SIAM Journal of Computing*, 28(2):652–673.

Fieldsend, J. E., Everson, R. M., and Singh, S. (2003). Using Unconstrained Elite Archives

for Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, 7(3):305–323.

Floyd, R. W. (1962). Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345.

Fonseca, C. and Fleming, P. (1996). On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers. In Voigt, H.-M., Ebeling, W., Rechenberg, I., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN IV*, volume 1141 of *Lecture Notes in Computer Science*, pages 584–593. Springer Berlin Heidelberg.

Fonseca, C., Guerreiro, A., López-Ibáñez, M., and Paquete, L. (2011). On the Computation of the Empirical Attainment Function. In Takahashi, R., Deb, K., Wanner, E., and Greco, S., editors, *Evolutionary Multi-Criterion Optimization*, volume 6576 of *Lecture Notes in Computer Science*, pages 106–120. Springer Berlin Heidelberg.

Fonseca, C. M. and Fleming, P. J. (1995). An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary computation*, 3(1):1–16.

Förster, A. (2007). Machine Learning Techniques Applied to Wireless Ad-Hoc Networks: Guide and Survey. In *Proceedings of the Third International Conference on Intelligent Sensors, Sensor Networks and Information*, ISSNIP '07, pages 365–370.

Frederickson, G. N. (1991). Ambivalent Data Structures for Dynamic 2-Edge-Connectivity and k Smallest Spanning Trees. In *Proceeding of the Thirty-Second IEEE Annual Symposium on Foundations of Computer Science*, pages 632–641.

Fredman, M. L. and Tarjan, R. E. (1987). Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. *Journal of the ACM*, 34(3):596–615.

Ganesan, D., Govindan, R., Shenker, S., and Estrin, D. (2001). Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(4):11–25.

Gungor, V. C. and Hancke, G. P. (2009). Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches. *IEEE Transactions on Industrial Electronics*, 56(10):4258–4265.

Gurobi (2015). Gurobi Optimizer Reference Manual. Available at `http://www.gurobi.com`.

Hart, P., Nilsson, N., and Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100 –107.

He, S., Chen, J., Xu, W., Sun, Y., Thulasiraman, P., and Shen, X. (2010). A Stochastic Multiobjective Optimization Framework for Wireless Sensor Networks. *EURASIP Journal on Wireless Communications and Networking*, 2010:9.

Hershberger, J., Maxel, M., and Suri, S. (2007). Finding the k Shortest Simple Paths: A New Algorithm and its Implementation. *ACM Transactions on Algorithms*, 3(4).

Hoffman, W. and Pavley, R. (1959). A Method for the Solution of the Nth Best Path Problem. *Journal of the ACM*, 6(4):506–514.

Hughes, E. (2003). Multiple Single Objective Pareto Sampling. In *Proceedings of the 2003 IEEE Congress on Evolutionary Computation*, volume 4 of *CEC '03*, pages 2678–2684.

Hughes, E. (2007). MSOPS-II: A General-Purpose Many-Objective Optimiser. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, CEC '07, pages 3944–3951.

Ishibuchi, H., Tsukamoto, N., and Nojima, Y. (2008). Evolutionary Many-Objective Optimization: A Short Review. In *Proceedings of the 2008 IEEE Congress on Evolutionary Computation*, CEC '08, pages 2419–2426.

Islam, O. and Hussain, S. (2006). An Intelligent Multi-hop Routing for Wireless Sensor Networks. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops*, pages 239–242.

Jaffrès-Runser, K., Schurgot, M., Comaniciu, C., and Gorce, J.-M. (2010). A Multiobjective Performance Evaluation Framework for Routing in Wireless Ad Hoc Networks. In *Proceedings of the 8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, WiOPT '10, pages 113–121.

Jeong, J. and Ee, C. T. (2003). Forward Error Correction in Sensor Networks. *University of California at Berkeley, USA*.

Jiménez, V. M. and Marzal, A. (2003). A Lazy Version of Eppstein's k Shortest Paths Algorithm. In *Experimental and Efficient Algorithms*, pages 179–191. Springer.

Kamath, S. and Nasipuri, A. (2011). Integrated Load Balanced and Energy Aware Routing in Large Scale Wireless Sensor Networks. In *Proceedings of the Thirtieth IEEE International Performance Computing and Communications Conference*, IPCC '11, pages 1–8.

Karmarkar, N. (1984). A New Polynomial-time Algorithm for Linear Programming. *Combinatorica*, 4(4):373–395.

Knowles, J. (2005). A Summary-Attainment-Surface Plotting Method for Visualizing the Performance of Stochastic Multiobjective Optimizers. In *Proceedings of the International Conference on Intelligent Systems Design and Applications*, ISDA '05, pages 552–557.

Knowles, J. D. and Corne, D. W. (2000). Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172.

Kohno, R., Meidan, R., and Milstein, L. (1995). Spread Spectrum Access Methods for Wireless Communications. *IEEE Communications Magazine*, 33(1):58–67.

Kulkarni, R. V., Förster, A., and Venayagamoorthy, G. K. (2011). Computational In-

telligence in Wireless Sensor Networks: A Survey. *IEEE Communication Surveys & Tutorials*, 13(1):68–96.

Lenzen, C., Locher, T., Sommer, P., and Wattenhofer, R. (2010). Clock Synchronization: Open Problems in Theory and Practice. In *SOFSEM 2010: Theory and Practice of Computer Science*, pages 61–70. Springer.

Li, B., Li, J., Tang, K., and Yao, X. (2015). Many-Objective Evolutionary Algorithms: A Survey. *ACM Computing Surveys*, 48(1):13:1–13:35.

Liu, D., Pang, J., Zhou, J., and Peng, Y. (2012). Data-Driven Prognostics for Lithium-Ion Battery Based on Gaussian Process Regression. In *Proceedings of the IEEE Conference on Prognostics and System Health Management*, PHM '12, pages 1–5.

Lücken, C. v., Barán, B., and Brizuela, C. (2014). A Survey on Multi-Objective Evolutionary Algorithms for Many-Objective Problems. *Computational Optimization and Applications*, 58(3):707–756.

Madan, R. and Lall, S. (2006). Distributed Algorithms for Maximum Lifetime Routing in Wireless Sensor Networks. *IEEE Transactions on Wireless Communications*, 5(8):2185–2193.

Madden, S. R., Franklin, M. J., Hellerstein, J. M., and Hong, W. (2005). TinyDB: An Acquisitional Query Processing System for Sensor Networks. *ACM Transactions on Database Systems*, 30(1):122–173.

Magaia, N., Horta, N., Neves, R., Pereira, P. R., and Correia, M. (2015). A Multi-Objective Routing Algorithm for Wireless Multimedia Sensor Networks. *Applied Soft Computing*, 30:104–112.

Miller, R. E. (1963). Alternative Optima, Degeneracy and Imputed Values in Linear Programs. *Journal of Regional Science*, 5(1):21–31.

Nedjah, N. and Mourelle, L. d. M. (2015). Evolutionary Multi-Objective Optimisation: A Survey. *International Journal of Bio-Inspired Computation*, 7(1):1–25.

Nesterov, Y. and Nemirovsky, A. (1994). Interior-Point Polynomial Methods in Convex Programming. *Studies in Applied Mathematics*.

Paradis, L. and Han, Q. (2007). A Survey of Fault Management in Wireless Sensor Networks. *Journal of Network and Systems Management*, 15(2):171–190.

Pollack, M. (1961). The k-th Best Route Through a Network. *Operations Research*, 9(4):578–580.

Radi, M., Dezfouli, B., Bakar, K. A., and Lee, M. (2012). Multipath Routing in Wireless Sensor Networks: Survey and Research Challenges. *Sensors*, 12(1):650–685.

Rahat, A. A. M., Everson, R. M., and Fieldsend, J. E. (2014). Multi-Objective Routing Optimisation for Battery-powered Wireless Sensor Mesh Networks. In *Proceedings of*

the 2014 Conference on Genetic and Evolutionary Computation, GECCO '14, pages 1175–1182, New York, NY, USA. ACM.

Rahat, A. A. M., Everson, R. M., and Fieldsend, J. E. (2015a). Evolutionary Multi-Path Routing for Network Lifetime and Robustness in Wireless Sensor Networks. *Ad Hoc Networks (Submitted for review)*.

Rahat, A. A. M., Everson, R. M., and Fieldsend, J. E. (2015b). Hybrid Evolutionary Approaches to Maximum Lifetime Routing and Energy Efficiency in Sensor Mesh Networks. *Evolutionary Computation*, 23(3):481–507.

Raynal, M. (2013). *Distributed Algorithms for Message-Passing Systems*, chapter Distributed Graph Algorithms, pages 35–58. Springer Berlin Heidelberg, Berlin, Heidelberg.

Roberts, B. and Kroese, D. P. (2007). Estimating the Number of *s*-*t* Paths in a Graph. *Journal of Graph Algorithms and Applications*, 11(1):195–214.

Rodoplu, V. and Meng, T. (1999). Minimum Energy Mobile Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1333–1344.

Rosenberg, R. (1967). Simulation of Genetic Populations with Biochemical Properties (Doctoral Dissertation, University of Michigan). *Dissertation Abstracts International*, 2(8):7.

Schaffer, J. D. (1985). Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100, Hillsdale, NJ, USA. L. Erlbaum Associates Inc.

Schmidt, D., Berning, M., and Wehn, N. (2009). Error Correction in Single-hop Wireless Sensor Networks: A Case Study. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '09, pages 1296–1301, 3001 Leuven, Belgium, Belgium. European Design and Automation Association.

Shin, K. Y., Song, J., Kim, J., Yu, M., and Mah, P. S. (2007). REAR: Reliable Energy Aware Routing Protocol for Wireless Sensor Networks. In *Proceedings of he 9th International Conference on Advanced Communication Technology*, volume 1 of *ICACT '07*, pages 525–530.

Spielman, D. A. and Teng, S.-H. (2004). Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time. *Journal of the ACM*, 51(3):385–463.

Srinivas, N. and Deb, K. (1994). Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evoltionary Computation*, 2(3):221–248.

Stadler, W. (1979). A Survey of Multicriteria Optimization or the Vector Maximum Problem, part I: 1776–1960. *Journal of Optimization Theory and Applications*, 29(1):1–52.

Talbi, E.-G., Mostaghim, S., Okabe, T., Ishibuchi, H., Rudolph, G., and Coello Coello, C. A. (2008). *Multiobjective Optimization: Interactive and Evolutionary Approaches*, chapter Parallel Approaches for Multiobjective Optimization, pages 349–372. Springer Berlin Heidelberg, Berlin, Heidelberg.

Valiant, L. G. (1979a). The Complexity of Computing the Permanent. *Theoretical computer science*, 8(2):189–201.

Valiant, L. G. (1979b). The Complexity of Enumeration and Reliability Problems. *SIAM Journal on Computing*, 8(3):410–421.

Vinyals, M., Rodriguez-Aguilar, J. A., and Cerquides, J. (2011). A Survey on Sensor Networks from a Multiagent Perspective. *The Computer Journal*, 54(3):455–470.

Wang, Q. and Balasingham, I. (2010). Wireless Sensor Network - An Introduction. In *Wireless Sensor Networks: Application Centric Design*, page 492. InTech.

Wu, B. Y. and Chao, K.-M. (2004). *Spanning Trees and Optimization Problems*. CRC Press.

Xue, F., Sanderson, A., and Graves, R. (2006). Multi-Objective Routing in Wireless Sensor Networks with a Differential Evolution Algorithm. In *Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control*, ICNSC '06, pages 880–885.

Yahya, B. and Ben-Othman, J. (2009). REER: Robust and Energy Efficient Multipath Routing Protocol for Wireless Sensor Networks. In *Proceedings of the 28th IEEE Conference on Global Telecommunications*, GLOBECOM '09, pages 5482–5488, Piscataway, NJ, USA. IEEE Press.

Ye, W., Heidemann, J., and Estrin, D. (2004). Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506.

Yen, J. Y. (1971). Finding the K Shortest Loopless Paths in a Network. *Management Science*, 17(11):712–716.

Yetgin, H., Cheung, K. T. K., and Hanzo, L. (2012). Multi-Objective Routing Optimization Using Evolutionary Algorithms. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, WCNC '12, pages 3030–3034.

Yick, J., Mukherjee, B., and Ghosal, D. (2008). Wireless Sensor Network Survey. *Computer Networks*, 52(12):2292–2330.

Young, J. K. (2008). What a Mesh! Part 2-Networking Architectures and Protocols — Sensors. *www.sensormag.com*.

Yurur, O., Liu, C. H., and Moreno, W. (2015). Modeling Battery Behavior on Sensory Operations for Context-Aware Smartphone Sensing. *Sensors*, 15(6):12323–12341.

Zang, I. (1980). A Smoothing-Out Technique for Min-Max Optimization. *Mathematical Programming*, 19(1):61–77.

Zhang, Q. and Li, H. (2007). MOEA/D: A Multi-Objective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731.

Zhao, J. and Govindan, R. (2003). Understanding Packet Delivery Performance in Dense Wireless Sensor Networks. In *Proceedings of the First International Conference on Embedded Networked Sensor Systems*, SenSys '03, pages 1–13, New York, NY, USA. ACM.

Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., and Zhang, Q. (2011). Multiobjective Evolutionary Algorithms: A Survey of the State of the Art. *Swarm and Evolutionary Computation*, 1(1):32–49.

Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, ETH Zurich, Switzerland.

Zitzler, E., Brockhoff, D., and Thiele, L. (2007). The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration. In Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., and Murata, T., editors, *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 862–876. Springer Berlin Heidelberg.

Zitzler, E. and Künzli, S. (2004). Indicator-Based Selection in Multiobjective Search. In Yao, X., Burke, E., Lozano, J., Smith, J., Merelo-Guervós, J., Bullinaria, J., Rowe, J., Tiňo, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 832–842. Springer Berlin Heidelberg.

Zitzler, E., Laumanns, M., and Bleuler, S. (2004). A Tutorial on Evolutionary Multiobjective Optimization. In *Metaheuristics for Multiobjective Optimisation*, pages 3–37. Springer.

Zitzler, E. and Thiele, L. (1999). Multiobjective Evolutionary Algorithms: A Comparative Case Study and The Strength Pareto Approach. *Evolutionary Computation, IEEE Transactions on*, 3(4):257–271.