Terms and Conditions of Use of Digitised Theses from Trinity College Library Dublin

Copyright statement

All material supplied by Trinity College Library is protected by copyright (under the Copyright and Related Rights Act, 2000 as amended) and other relevant Intellectual Property Rights. By accessing and using a Digitised Thesis from Trinity College Library you acknowledge that all Intellectual Property Rights in any Works supplied are the sole and exclusive property of the copyright and/or other IPR holder. Specific copyright holders may not be explicitly identified. Use of materials from other sources within a thesis should not be construed as a claim over them.

A non-exclusive, non-transferable licence is hereby granted to those using or reproducing, in whole or in part, the material for valid purposes, providing the copyright owners are acknowledged using the normal conventions. Where specific permission to use material is required, this is identified and such permission must be sought from the copyright holder or agency cited.

Liability statement

By using a Digitised Thesis, I accept that Trinity College Dublin bears no legal responsibility for the accuracy, legality or comprehensiveness of materials contained within the thesis, and that Trinity College Dublin accepts no liability for indirect, consequential, or incidental, damages or losses arising from use of the thesis for whatever reason. Information located in a thesis may be subject to specific use constraints, details of which may not be explicitly described. It is the responsibility of potential and actual users to be aware of such constraints and to abide by them. By making use of material from a digitised thesis, you accept these copyright and disclaimer provisions. Where it is brought to the attention of Trinity College Library that there may be a breach of copyright or other restraint, it is the policy to withdraw or take down access to a thesis while the issue is being resolved.

Access Agreement

By using a Digitised Thesis from Trinity College Library you are bound by the following Terms & Conditions. Please read them carefully.

I have read and I understand the following statement: All material supplied via a Digitised Thesis from Trinity College Library is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of a thesis is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form providing the copyright owners are acknowledged using the normal conventions. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

Distributed Optimal Routing in Wireless Ad Hoc Networks

Guoxian Yang

A Dissertation submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

December 2012

Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work.

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this Dissertation upon request.

Acknowledgements

Years ago, when I attended the M.Sc course in Ubiquitous Computing, I learned that the only criterion for Trinity to grant a Ph.D degree is the candidates contribution to the knowledge. This very idea excited and motivated me to start my Ph.D life in the Distributed System Group.

Over the years, I have the freedom to explore different domains of computer science and worked on a few – in my opinion interesting yet difficult problems. At the conclusion of my Ph.D course, I cannot say that I have accomplished much, but I am truly happy with the mind I have developed.

Pursuing a PhD is a long and sometimes lonely journey. I am grateful to all the people accompanied me through it. My special appreciation goes to my special one, my wife Yang Yang, who is always there for me and backs me up whenever I feel desperate. To my supervisor, Dr. Stefan Weber, thank you for guiding me through the journey. As much as your knowledge in computer science, I respect your dignity and honesty as an academic. To Tim John Walsh and Andronikos Nedos, thank you for welcoming me as your friend since the start of my Ph.D. You guys are like my older brothers that I never have. To Dominik Dahlem, thank you for so many inspiring conversations, which I hope will continue to happen after my academic life. To Serena Fritsch, thank your for always being supportive. To Jan Curn and Asad Salkham, your special sense of humor helped to cultivate my humor. To Luca Longo, I will remember all the basketball nights when we sweat our trouble out the court. To Mithileash Mohan, you are the best travel,

basketball, alcohol-drinking (although you do not), and coffee-drinking companion. To Brendan Cody Kenny, David Guerin and Michael Clear, I always enjoyed our random discussions in our office. My appreciation also goes to Lawrence Lee, for your advices on becoming an entrepreneur and to Zhou Dong, for your advices on becoming an academic. Finally, to my parents, Han Fang and Yang WeiZhen, for support my dream.

I hope the curiosity never dies.

Guoxian Yang

University of Dublin, Trinity College

December 2012

Abstract

In this thesis, optimal routing problems in wireless ad hoc networks are investigated. The majority of existing routing algorithms follow a selfish strategy, where routing decisions are based on individual communication flows. The performance of these algorithms may suffer from the competition between multiple communication flows, also known as the tragedy of the commons. This issue worsens in wireless ad hoc networks because communications in these networks experiences more mutual interference between flows in comparison to communication in wired networks.

Optimal routing approaches apply results of optimization theory such as the gradient descent method or the Newtons method to network routing problems. These approaches optimize the overall performance of all communications in a network with respect to certain criteria; in this thesis, I chose the delay experienced by all flows as the optimization criterion.

The limitations of existing optimal routing approaches are threefold. Firstly, they model the routing problem as a link graph, where each vertex represents a router and each link represents a connection between two routers. This link model suits wired networks where the main factor influencing communication over a link is the existing traffic over this link. However, in wireless communications, a transmission between two neighbouring nodes is not only subject to existing traffic between these nodes but also to interfering traffic in the vicinity. Secondly, existing optimization methods coordinate routing decisions of all communication flows which has been realized in centralized

algorithms. Distributed optimal approaches rely on significant amounts of transfers of information to support these optimizations, which does not suit wireless networks, where the bandwidth is limited. Finally, existing approaches assume the availability of all potential paths in a network and focus on the data distribution over them. However, the number of potential paths between a source and a destination node may be very large. It may be practically infeasible to discover and to propagate the cost information of all possible paths in this scenario.

In this thesis, I propose a function called the wireless medium cost that models the latency incurred by wireless interference. It is proven that the routing problem in wireless networks is indeed convex. Therefore, many strong results in optimization theory can be applied to this problem. Using the wireless medium cost, I propose a fully distributed optimal routing algorithm for wireless ad hoc networks. It is then demonstrated that the result of this algorithm is significantly better than that of a selfish algorithm, in the sense that the lower bound of the cost of selfish routing divided by the cost of optimal routing is infinite. As the evaluations confirm, our proposed algorithm converges significantly faster compared to existing distributed optimal approaches. These theoretical results are then incorporated in the design of a novel route discovery method that discovers loop-free necessary paths used by the optimal communications approach.

Contents

Bibliography **223**

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Context

A wireless ad hoc network is an infrastructure-less network where preexisting centralised control may not be present. Devices, termed as *nodes*, within transmission range of each other are interconnected through wireless communication. Although it is possible to organise a hierarchical topology where a number of clusters are organized, we focus our study on a flat topology where each node acts as a router and relays data traffic for other nodes.

Unicast routing is one of the fundamental tasks in the networking domain. It addresses the discovery and maintenance of forwarding paths between a source node and a destination node. Some of the well established unicast routing approaches are as follows. *Shortest path routing* protocols, such as AODV, OLSR, DSR, etc., aim to set up a path with the least hop count between source and destination nodes. These protocols are simple to implement but overlook the traffic dynamism of a network that may result in hot spots of congested nodes. Many adaptive approaches have been proposed to avoid highly congested paths. In this category, *minimum latency routing* protocols searches for a path with least latency for a communication flow; *Least transmission routing* protocols

take the retransmission incurred by interference into account; Queue length-based metrics and number of interfering nodes have also been used as criteria of route discovery. In order to provide robustness and reliability to data transmission, multipath routing protocols establish multiple disjoint paths for a single communication flow. All these approaches follow a selfish strategy in that each communication flow routes its data traffic through single or a set of route(s) with the least cost under certain metrics, e.g. hop counts, latency, etc. The performance of such a selfish approach may suffer from the competition between different flows, also known as the "tragedy of the commons".

An optimal routing approach is one that optimizes the behaviour of all communication flows in a network. It can be formulated as an application of optimization theory: Given the traffic demand of all communications in a network, D, and a cost function of the network, $C(\cdot)$, an optimal routing protocol discovers potential paths and a feasible traffic distribution on the paths, F^* so that the overall cost of the network is minimized, $C(F^*) = \min. \{C(F)|\forall \text{F is feasible}\}$. Under many objectives of practical interests, such as latency experienced by network traffic, power consumed by all the nodes, etc., the cost function $C(\cdot)$ can be modelled as or modified into an increasing convex function, where strong results from optimization theory reside. According to the Karush-Kuhn-Tucker (KKT) condition, under an optimal solution, F^*, for any communication, the marginal costs of all used paths are the same and are no greater than that of any unused paths. This state is also known as the optimal Wardrop equilibrium.

One class of methods that is commonly used in existing optimal routing protocols is the gradient-based method. The marginal or the first derivative cost of each path, is used as the cost metric. Each communication iteratively increases the traffic on paths with smaller marginal cost and decreases the traffic on the paths with larger marginal cost until the optimal Wardrop equilibrium is reached. These methods are simple to implement in comparison to other optimization approaches but suffer from a slow convergence rate. A more mathematically advanced class of methods is the second order Newton's method [Boyd and Vandenberghe, 2004]. In addition to the first

derivatives, it uses the second derivatives, i.e. the Hessian matrix of the cost function, to speed up the convergence to optimum.

The problem of finding an optimal solution in wireless ad hoc networks, although lacking of research efforts, is of particular interests. As the physical processing power of wireless devices increases, the future applications of wireless ad hoc networks may see substantial demands of data traffic. The communication resources, however, are scarce in ad hoc networks: In comparison to wired transmission, wireless transmission experiences lower data rate and is error prone; Ad hoc devices may have limited power supply. Therefore, ad hoc networks thirst for approaches that improve the performance of communication. At the same time, an ad hoc network lacks central control. More importantly, wireless signals transmit on an open medium. Neighbouring communication flows will compete for medium access and interfere with each other's transmission. In such highly coupled systems, selfish approaches are likely to suffer more from anarchy. It gives the space for an optimal routing solution to improve networking performance from selfish approaches.

1.2 Problem Domain and Notations

In this section, we firstly define the goal and the problem domain of this thesis. Then, we describe some of the basic notations in this thesis.

1.2.1 Objectives

Common cost metrics of network optimization include traffic delay, power consumption, throughput, and etc. We choose the communication delay experienced by all the data traffic in a network as the cost metric of our routing optimization. This choice of the cost metric is motivated as follows: 1) The communication delay is an important criterion to applications; 2) It is a good indicator of network congestions, which will leads to deterioration of many other performance metric, such as data delivery ratio; 3) It is

sensitive to routing behaviours. Bad routing decisions will introduce large delays by either creating long forwarding paths or causing congested nodes. Therefore, in this thesis, we focus on challenges and approaches to minimize the cost of delay, although a large portion of our solution can be transferred to optimize other forms of cost metrics.

1.2.2 Scope and Assumptions

A routing task solves the assignment of paths by which data traffic are forwarded. For a single path routing algorithm, it involves the discovery of the best path based on certain criteria. In an optimal routing algorithm, multiple paths may be utilized by a communication flow, as long as the overall network cost is minimized. The scope of an optimal routing algorithm is slightly wider than single-path routing algorithms. The functionalities that our optimal routing algorithm covers are listed below:

1. the discovery of multiple potential paths
2. the decisions of a subset of paths to use
3. the data distribution on multiple paths

Throughout the thesis, we assume an arbitrarily splittable traffic unless stated otherwise. In practice, this can be achieved by the fragmentation of data packets along forwarding paths, although it will lead to large overhead. A more popular method is to approximate it with weighted round robin at intermediate nodes.

We focus on scenarios where a network sees a large and persistent traffic demand so that congestion exists. A practical example can be large file sharing between node pairs.

We assume a quasi-static network topology in the sense that we do not address the mobility of nodes explicitly but aim to achieve a fast convergence of our routing algorithm in order to react to topological change.

We further limit our attention to single communication channel for wireless transmission. Neighbouring transmissions share the medium in a time-division manner. That is,

a node can only correctly receive one communication signal at any time slot. We assume a perfect transmission in the sense that if two nodes are within each other's transmission range, they can transmit data to each other. The cost incurred by contention of medium access and by retransmission due to collision are modelled by a cost function.

Finally, we do not consider the battery life of wireless nodes and assume all nodes have infinite battery life.

1.2.3 Notations

In this section, we present definitions and notations that are used throughout the thesis. A network $\mathcal{G} = \{\mathcal{V}, \mathcal{L}\}$ is defined as a set of nodes $\mathcal{V}$ and a set of wireless links $\mathcal{L}$. A communication pair $w = \{s, d\}$ includes a source node s, and destination node d. A path $p \subset \mathcal{L}$ is a set of connected links. $\mathcal{P}$ denotes all paths in the network and $\mathcal{P}_w$ denotes all paths between a communication pair w.

We denote the traffic demand for communication w as $\mathcal{T}_w$. A flow f_p denotes the data that are transmitted over a path $p \in \mathcal{P}_w$. Let $\mathcal{F} = \{f_p | p \in \mathcal{P}\}$ denote the overall flow pattern of a network. We call $\mathcal{F}$ a feasible solution to a routing task if for any w, $\mathcal{T}_w = \sum_{p \in \mathcal{P}_w} f_p$.

The data load $\bar{f}_a$ of node a, is the summation of all flows that run through it:

$$\bar{f}_a = \sum_{p \in \mathcal{P}} \delta_{ap} f_p \tag{1.1}$$

where

$$\delta_{ap} = \begin{cases} 0 & \text{if } a \notin p \\ 1 & \text{if } a \in p \end{cases}$$

We define $\mathcal{N}_a$ as the set of neighbours of node a, and accordingly define N_a as the neighbouring traffic of node a:

$$N_a = \sum_{x \in \mathcal{N}_a} \bar{f}_x$$

Communication delay is the result of underlying transmission ability, interference and workload over a medium. We define a latency function $\ell(\cdot)$ over one hop. It takes

the data load of the medium as input. We assume the latency function to be convex increasing. We define the cost function over one node, C_a, as the latency experienced by all communications that run through the node.

$$C_a = \ell(\bar{f}_a + N_a)\bar{f}_a$$

The cost of a network is given accordingly.

$$C(\mathcal{F}) = \sum_{a \in \mathcal{V}} C_a$$

Given a network and the traffic demand, the objective of our solution, namely the network optimum, is to find a feasible $\mathcal{F}$ that minimizes the network cost.

$$\underset{\mathcal{F}}{\text{minimize}}\ C(\mathcal{F}) \tag{1.2a}$$

$$\text{s.t.} \sum_{p \in \mathcal{P}_w} f_p = \mathcal{T}_w \tag{1.2b}$$

$$\mathcal{F} \succeq 0 \tag{1.2c}$$

$$g(\mathcal{F}) < \mathcal{C} \tag{1.2d}$$

1.3 Challenges

Wireless ad hoc networks impose new challenges to the design of optimal routing protocols. In this section, we discuss some of the difficult issues for wireless optimization, which we will address later through the rest of this thesis.

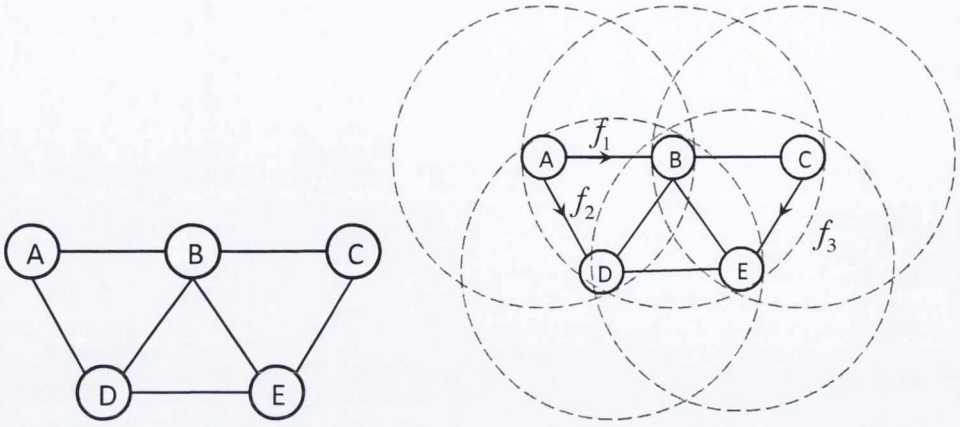
1.3.1 Wireless Cost Function

Optimal approaches use marginal cost - the derivative of latency in our case, to make routing decisions. During the runtime of communications, the value of the derivatives can not be measured but only inferred from the change of latency measurements. Such an estimate suffers from low accuracy and deteriorates further if one wishes to use the second derivatives to speed up the convergence. Therefore, unlike selfish approaches,

optimal approaches require an explicit cost function that models the curvature of latency value with satisfying accuracy, of which both the first and the second derivatives can be computed during algorithm runtime.

Due to the broadcast nature of radio signals, the wireless interference in ad hoc networks differs from the link-based interference in wired networks. In the link-based model, the latency experienced by data traffic from one node, say A, to another node, say B, is only subject to the amount of traffic along the link AB. Combining all links and nodes, a network can be represented by a weighted graph, for example one shown in Fig.(1.1). With full or partial knowledge of the graph, a wired routing algorithm, be it optimal or selfish on a given cost metric, makes routing decisions. On the other hand, in ad hoc networks, the latency experienced by data traffic depends on traffic of multiple links that contend for medium access. For example, in an ad hoc network with the same topology shown in Fig.(1.1), the cost of traffic from A to B depends on the outgoing and incoming traffic of nodes A, B and D, as well as the incoming traffic of nodes C and E. Therefore, existing link-based cost model and weighted graph proposed for wired networks do not suit wireless communication.

The analysis of the wireless interference for the design of a cost function is non-trivial. Let us first examine this issue at the medium access control (MAC) layer. Different interfering traffic may have different mechanisms of impacts on the cost of one traffic link. In the example of Fig.(1.1), the link traffic $f_1$is interfered by traffic f_2 through joint queueing and shared medium access at node A, while it is interfered by traffic f_3 through medium contention. In addition, the two types of interference may be coupled. Traffic f_3 impacts the performance of traffic f_2 through its interfering with the medium access of traffic f_1. Moving on to the network layer, we can see that different traffic flows may interfere with each other at multiple occasions with different types of impact. In fact, exclusive to the domain of ad hoc networking, a traffic flow interferes with itself as adjacent intermediate nodes of a path contend for medium access. It complicates the design of a cost function with regard to network traffic flow.



(a) A wired network (b) A wireless network

Fig. 1.1: View of Interference: Wired v.s. Wireless

In summary of this section, it is both necessary and challenging to derive a close-form cost function for optimal ad hoc routing algorithms. It is also interesting to note that, because of the complicated interference model, one shall not assume the convexity of the networking problem without evidences.

1.3.2 Distributed Methods

Wireless ad hoc networks postulate a distributed type of routing algorithms, which is a challenge for optimization approaches.

The distributed manner of a routing algorithm can be understood from two aspects. From the computation aspect, routing decisions of each node, i.e. the distribution of traffic to next hop nodes, should be made locally. For an optimal routing, this involves decomposing an optimization task into multiple components, which are carried out at different nodes. From the communication aspect, information exchanged between remote components should be kept at a minimum. Considering that the rationale behind optimal routing approaches is for different communication flows to behave in coordination rather than being selfish, it is in general a challenge to design an optimal routing algorithm with a high degree of decoupling between components.

1.3.3 Route Discovery

The issue of route discovery for optimal routing is essential but missing for ad hoc networks. Because nodes may transmit data to all their neighbours, a populated ad hoc network may have a high degree of connectivity. There may be an astronomical amount of potential paths between a source node and a destination node. Existing optimal routing approaches mainly assume the presence of all potential paths and compute the data distribution on all the paths, although not all of the paths may be used. While this is feasible in a backbone network as routers can be known a priori and remain static, it is infeasible to discover and maintain a large amount of paths in ad hoc networks. Existing multipath route discovery approaches aim to establish a small number of disjoint paths. Although the criterion of disjointness shows evidences of robustness, the optimality of discovered paths can not be guaranteed.

1.4 Proposed Solutions

1.4.1 Wireless Medium Cost

Our journey to an optimal routing algorithm starts with an investigation of cost of latency in wireless ad hoc networks. We introduce a cost function, called wireless medium cost, to replace the link cost in wired communication. It captures the latency incurred by a network traffic flow upon traffic on the disc of its radio coverage. The key is to model the delay at each node through a multi-class first come first serve (FCFS) queue with general service. Each class of traffic represents packets addressed for a certain station therefore experiences different service. The mean and second moment of the service time - the medium access time plus the transmission time - of all classes are the input of the queueing model. They can be calculated separately using results from polling systems or analytical model for specific MAC protocols. Finally, we arrive at the wireless medium cost by summing all queueing and service delay factored by their traffic flow. Because

the wireless medium cost models the impacts of a traffic flow, the marginal medium cost of a path is the derivative of the network objective cost to the traffic along the path.

With cost function defined, we discuss the convexity of the wireless optimal routing problem and demonstrate that the problem is in fact strictly convex if the medium access time is convex with respect to data rate.

1.4.2 A Distributed Optimization Method

We propose an optimal routing approach using the wireless medium cost function to address the problem (1.2). The core of the approach is a quota-based mechanism integrated with the interior-point method.

The interior-point method is an extension to the classical Newton's method in order to address optimization problems with inequality constraints. It introduces an extra cost for the violation of inequality constraints by adding a barrier function to the objective cost function. The barrier function increases to infinity when approaching infeasible regions of the inequality constraints. Thus it keeps Newton's direction to feasible regions. It has been shown that the interior-point method retains the superlinear convergence rate of Newton's method. In the domain of optimal routing, the barrier function takes the summation of competing traffic flow at each hop - upper bounded by the rate capacity - as its input. One barrier per hop is added a path cost. Therefore, through the barrier method, multiple communication flows are heavily coupled. It results in difficulties for distributed implementation of the algorithm.

In order to address this issue, we develop a quota-based mechanism. Each node calculates its local medium capacity residual, taking its neighbouring traffic into account. Note that neighbouring nodes may have different views of capacity residual. According to some rules that guarantees the feasibility, each node assigns a share of the medium capacity residual, namely a quota, to each traffic flow that runs through it. The quota of a path is the minimum one from all its intermediate nodes. The barrier function takes the quota value of a path as its input. During the computation of Newton step,

one barrier per path is added. The quota-based mechanism release the coupling among communication flows. We demonstrate that it is more efficient compared to the standard distributed implementation of interior-point method.

The quota-based mechanism distributes the optimization task into different communication flows. Next, we explore the substructure property of our routing approach and reform it from source routing to destination routing. This reformation can be roughly understood as distributing the optimization task of one communication flow into its intermediate nodes.

To the best of our knowledge, existing Newton-based routing approaches can all be classified as source routing: A source node gathers the cost information of all its paths to each destination node and computes its routing decisions. This creates communication overhead and imposes heavy computation loads on source nodes.

Similar to the well-studied single path destination routing, we decompose our routing approach iteratively into subtasks of routing from intermediate nodes to destination nodes. This procedure can be understood from two directions. In destination-to-source direction, cost information and quota values of multiple paths merge at their intersection node. We devise rules of the merge so that the resulting routing solution routes the same amount of traffic to the intersection node as if paths are not merged. In source-to-destination direction, given its receiving traffic, either from application layer or from last hop nodes, each node calculates a traffic assignment to next hop nodes according to their cost information and quota-value.

In summary, we propose a node-level distributed quota-based interior-point method to address the problem (1.2).

1.4.3 Route Discovery

We investigate issues of on-demand multipath route discovery for optimal routing, although many results can be translated to a proactive manner. In a route discovery process, each source node floods a route request message before data transmission. Route

tables are updated by route reply messages from destination nodes.

Our study in route discovery can be summarized by three parts. Firstly, we take advantages of wireless transmission and establish some propositions exclusive in ad hoc networks to prune the flooding of route request messages. Loops and paths that contain other paths are removed during the forwarding of request messages. Secondly, we realise that the update of routing tables for an optimal destination routing requires special care. For a cost of latency, the merge of cost from multiple paths results in a smaller cost than that of individual paths, which leads to loops in per-hop forwarding. We develop a mechanism that separates the concepts between costs of next hops and orders of intermediate nodes in a path. Finally, it turns out that a by-product information of Newton's method, the dual direction, can be used as a time-to-live (TTL) to filter out 'unnecessary paths' during route discovery. By 'unnecessary paths' we mean paths that will be used in the optimal routing solution of our approach. This enables our routing algorithm to start a small set of paths - even the shortest single path in hop counts - and to expand quickly to the optimal set of paths after a few Newton steps in data transmission.

1.4.4 Diagonal Hessian Approximation

The computation of Newton step involves the inverse of a Hessian matrix of the objective function, which is known to be computationally expensive to carry out. The standard approach is to introduce an iterative method, such as Jacobi method, Gauss-Seidel method or preconditional conjugate gradient method, to compute each Newton step. We argue that these nested iterations transform the computational complexity of the problem to the communicational complexity of the problem, therefore do not suit ad hoc networks.

Our optimal routing approach is established on a diagonal approximation of the Hessian matrix. Such an approximation reduces largely the complexity of the optimization at the price of a slower convergence rate. We demonstrate statistically that in most

scenarios of our simulations, the convergence rate is still satisfying.

1.5 Structure of the Thesis

The thesis is organized as follows. In chapter 2 and 3, we review the state of the art results in routing algorithm domain and in theoretical optimization domain respectively. By the end of each chapter, we summarize open questions in that domain for the design of an optimal routing algorithm. In chapter 4 we present our design of a wireless medium cost that models wireless interference. In chapter 5, we give the design of an optimal routing algorithm using the wireless medium cost to optimization wireless communication. In chapter 6, we verify the fundamental arguments of our approach. In chapter 7, we conclude the thesis and point the future direction.

Part I

State of the Art

The motivations and the challenges of our proposed solution to the wireless optimal routing task can not be fully revealed or justified without a scrutiny of our knowledge in related domains. In this chapter, we review some of the state of the art that either are closely related to our solution or presents tools that we leverage in the rest of the thesis.

Firstly, we review some of the classical routing algorithms. The term 'classical' here refers to routing approaches that do not consider the theoretical optimization. Most of these approaches follow a selfish strategy. However, they provide engineering insights and solutions to issues in routing domain which also exist in optimal routing domain.

Secondly, we review existing optimal routing algorithms, which are close competitors of our solutions. We start by giving an overview of the system model that the majority of the optimization approaches follow. Then, we categorize optimal algorithms by their mathematical methods. In each category, we present its mathematical background and discuss routing algorithms using the method.

Finally, we review recent progress in the delay analysis of wireless communication. It includes results from polling systems, MAC layer scheduling and modelling of 802.11 DCF. These results can be used in our development of a cost function in next chapter.

Chapter 2

Classical Routing Approaches

The majority of existing routing algorithms do not optimize an objective function of a network but rather they establish paths with the least given metric for each communication. They focus on removing loops in paths, reducing the control overhead and avoiding instability of forwarding paths. In this section, we review these classical routing approaches whose results may be carried on to the design of optimal routing algorithms.

2.1 Link-State Routing

Routing algorithms, by the form of message exchange, are commonly classified into link-state routing and distance-vector routing. The optimal routing algorithm we propose can be classified as a distance-vector. In the following, for the sake of the completeness of the review, we start briefly from the less related link-state routing before discussing distance-vector routing in next subsection.

A link-state routing algorithm can be roughly characterized as 'each node telling the network about its neighbouring links information'. Through link sensing, each node discovers its neighbours and the cost metric of the link to each neighbour, i.e. the link-state information. Each node periodically floods the link-state information advertisement to all nodes in a network. Receiving advertisements from all other nodes, each node can

reconstruct a network map. The Dijkstra's Shortest Path Algorithm (SPA) [Dijkstra, 1959] can be used locally to compute the shortest path to all destination nodes.

Open Shortest Path First (OSPF) [Moy, 1998] is a link-state routing protocol that has been widely deployed in enterprise networks. Another link-state routing protocol, namely Intermediate-System to Intermediate System (IS-IS) [Oran, 1990], is commonly used in ISP networks. In ad hoc networks, Optimized Link-State Routing protocol (OLSR) [Clausen and Jacquet, 2003] has gained a wide popularity. The flooding of the link-state information can be expensive in overhead. A simple technique to reduce such a flooding is for nodes to keep track of the link-state advertisements and to avoid sending multiple copies of the same one. Some more advanced flooding reduction techniques include area routing [Moy, 1998], fish-eye state routing [Pei et al., 2000], approximate link-state routing [Levchenko et al., 2008] and multi-point relays (MPR) [Clausen and Jacquet, 2003]. Other than the MPR technique that utilizes the broadcast nature of wireless transmission, most of these techniques reduce the flooding overhead at a cost of losing accuracy.

Despite these flooding reduction techniques, link-state routing can be still expensive in control overhead for ad hoc networks. The core issue of link-state routing is that in a flat, structureless network such as an ad hoc network, every node needs to maintain a global knowledge of the network topology. It in general requires that each node periodically floods its link information. This means that at each node, paths are maintained to all other nodes at all time, which are unnecessary. To the best of our knowledge, only a handful of exceptions exist. Notable proposals include: Source-tree On-demand Adaptive Routing (SOAR) [Roy and Garcia-Luna-Aceves, 2001] and On-demand LInk Vector Routing (OLIVE) [Garcia-Luna-Aceves and Roy, 2005]. They allow a node to discover part of the network topology to compute paths to a destination node on the event of data arrivals. The trick is, instead of sharing state information of neighbouring links, nodes share state information of all links in the path to a requested destination. This mechanism resembles much similarities to on-demand distance-vector routing, be-

cause it is the path information that are shared between neighbours. Further studies are required to distinguish and compare these two types of routing algorithms.

In summary, the large amount of control overhead is not desirable in wireless ad hoc networks where our interest lies. Therefore, we design our optimal routing algorithm in a on-demand distance-vector form. However, we do not eliminate the possibility of a link-state implementation of our optimal routing nor do we project the comparison between the optimal distance-vector routing and the potential optimal link-state routing.

2.2 Distance-Vector Routing

A distance-vector routing algorithm can be roughly characterized as 'each node telling its neighbours the shortest path it has to known destinations'. In contrary to link-state routing, nodes do not try to construct the network map. But rather, each node maintains a routing table with the least distance to each known destination and the next hop neighbours through which it can reach each destination. Neighbouring nodes share their routing information. In this way, best routes towards a destination node is gradually built from its neighbours to remote nodes. This is method is called distributed Bellman-Ford algorithm [Bellman, 1957,L. R. Ford and Fulkerson, 1962]. Notable distance-vector routing protocols include Routing Information Protocol (RIP) [Malkin, 1998], Interior Gateway Routing Protocol (IGRP) [IGR, 1991] and Enhanced Interior Gateway Routing Protocol (EIGRP) [Albrightson et al., 1994] in wired networks.

One desirable feature of distance-vector routing in ad hoc networks is that it is straightforward to implement in an on-demand manner. A general approach is as follows. On receiving a data packet addressed for an unknown destination, a node generates and sends out a route request message (RREQ) to all its neighbours. If a neighbour node has a route to the destination node, it generate a route reply message (RREP) with a distance. Otherwise, it forwards the RREQ message further to its neighbours. If none of the network nodes have stored routes to the destination, the RREQ message will be

forwarded and eventually reach the destination node. It send a RREP message back through the forwarding path of its corresponding RREQ message. The RREP message increases its distance at each hop. On receiving a RREP message, a node updates its routing table with the next hop neighbour with the least distance to the destination. Data messages are sent through the shortest path. Examples include Ad hoc On-demand Distance-Vector routing (AODV) [Perkins et al., 2003] and dynamic MANET on-demand (AODVv2) routing [Perkins et al., 2012]. In next section, we examine an infamous problem of distance-vector routing and review solutions for this problem.

2.3 Loops and Count-to-Infinity Problem

The fact that nodes do not maintain a global knowledge of a network topology has a downside. Distance-vector routing suffers a so-called count-to-infinity problem. The problem arises from mutual deceptions of neighbouring nodes. When network topology changes, loops may occur between neighbouring nodes. For example, as shown in Fig.(2.1), node A and B can reach node E through C and D. When link CD breaks, node C will falsely believe that through A and B it can reach node D. The three nodes continuously point to each other for next hop. Hop counts at each node increase to infinity. Loops are formed.

Existing methods to address the count-to-infinity problem can be classified into four categories: limits on hop count, use of sequence number, diffusing computation, and path finding. In the following, we briefly explain these methods.

2.3.1 Hop Count Limits

A simple approach to eliminate the count-to-infinity problem is to set a upper bound on hop counts for each node. Once the limit is reached, a node treat the link as broken. After a number of rounds, the loops are removed. This approach suffers from a slow convergence rate. Some improvements have been developed on top of this method. For

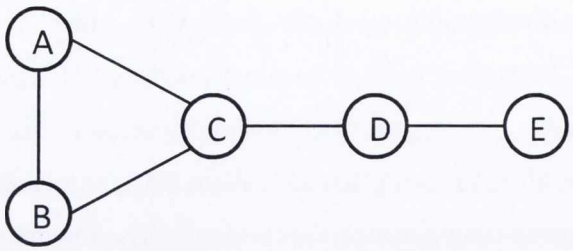


Fig. 2.1: Count-to-Infinity Example

example, RIP uses a split horizon with poison reverse technique [Malkin, 1998]. But they do not fully eliminate the problem.

2.3.2 Destination Sequence Number

Sequence number has been used to indicate the freshness of information at the network layer. Perkins and Bhagwat proposed to use sequence number to remove loops in their Destination-Sequenced Distance Vector (DSDV) [Perkins and Bhagwat, 1994] routing protocol. The use of sequence number has been adopted in AODV and AODVv2. The idea is to attach an incremental sequence number with information originated from a destination node. For the example in Fig.(2.1), let us assume the sequence number of node D is 5, which is known for all nodes. Once link CD breaks, node C attaches an increased sequence number 6 to a RREQ message. On receiving this message, node A and B invalidate their route information as they are out of dated. Assuming node B has a new separate route towards D, the RREQ message will be forwarded to node D through that path. Node D replies a RREP message with an increased sequence 7, which will update all routing information at node A, B and C.

2.3.3 Path-Finding

One angle to view the count-to-infinity problem is that nodes do not have topology information. After link changes in a network, nodes need to recalculate routes. When a node receives new information about a route to a destination node from its neighbour,

it has no knowledge about whether or not it is already somewhere on the route from the neighbour to the destination. Therefore, loops may occur. Following this rationale, an intuitive solution is for nodes to learn about links of their known paths.

Path-finding algorithms are a type of distance-vector routing that help nodes to learn topology information efficiently. In addition to distance information, messages exchanged between neighbours also include predecessors on paths. A predecessor of a path towards a destination node is the second last hop of the path. The predecessor and the destination node of a path together constitute the last link of the path. From paths information to all known destinations, a node can construct the topology of a network and use it to prevent loops. For example, in the network shown in Fig.(2.1), node A receives from node C path information towards node D and E and from node B path information towards node C, D, and E. Node A can construct the topology of the network. In case, link (C, D) breaks. Node A can rule out paths from both node C and B, because it understands that the path through B include node C.

A path-finding algorithm shares some similarity with a link-state routing as in both cases, network topologies are construct at each node. However, unlike link-state routing where link-state information is flooded, in a path-finding algorithm remote links are shared between neighbours together with path distances. Nodes choose next hops based on their reported distance to a destination. Topology information is used only to detect loops. It has been argued that path-finding algorithms outperformed proactive link-state algorithms [Garcia-Luna-Aceves and Murthy, 1997, Vutukury and Garcia-Luna-Aceves, 2000a]. Examples of path-finding algorithms includes Loop-free Path-finding Algorithm (LPA) [Garcia-Luna-Aceves and Murthy, 1997] and an algorithm proposed by Humblet [Humblet, 1991]

In the event of a topology change, temporal loops may occur in path-finding algorithms before predecessor information of all intermediate nodes is updated.

2.3.4 Feasibility Condition

Similar to the use of sequence number in DSDV and AODV, the feasibility condition is a mechanism to detect and prevent potential loops at every instant. In order to understand the rationale of the feasibility condition, we can make a simple observation from the view of a data packet.

Imagine that we are going from a start point to a destination point. Regardless of the routes we take, as long as each step results in a strictly decreasing distance towards the destination, we can not travel in circles. Topology changes in a network may leave false information at intermediate nodes. Following false road signs, we may violate the distance descent requirement and end up in loops.

The feasibility condition introduces a new distance value called feasible distance between two nodes i and j, denoted by FD^i_j. A feasible distance is a copy of the historical minimum value of its distance, D^i_j, to destination j, since the last time it was safely assigned. Denote the distance reported from neighbour n to i towards destination j as D^i_{jn}, which is essentially a copy of D^n_j but is noted differently to indicate that it is a view of node i. Denote the link distance between neighbours i and n as $l_{i,n}$. Denote the next hop node from node i to destination j, the successor, as s^i_j. It states that [Garcia-Luna-Aceves and Murthy, 1997]:

"If at time t router i needs to update its current successor, it can choose as its new successor s^i_j *any router* $n \in N_i$ *such that 1)* $D^i_{jn} + l_{in} = Min\{D^i_{jx} + l_{ix} | x \in N_i\}$ *and 2)* $D^i_{jn} < FD^i_j$*"*

The first condition indicates a selection of the shortest path. Because $D^i_j > FD^i_j > D^i_{jn} = D^n_j$, the second condition guarantees that the actual distance can only be decreasing at each hop. Using the historical minimum distance as a guard, the feasibility condition is conservative. It has been proven to be a sufficient but not necessary condition for loop-freedom at every instant [Garcia-Luna-Aceves and Murthy, 1997].

If a node can not find a successor that meets the feasibility condition, it initializes a query for routes to the destination node. Diffusing computations [Dijkstra and Scholten,

1980] is a signalling scheme to avoid false terminations for distributed computation among multihop nodes. Diffusing Update ALgorithm (DUAL) [Garcia-Lunes-Aceves, 1993] makes use of the diffusing computation technique to propagate route queries for the feasibility conditions. EIGRP [Albrightson et al., 1994] is based on DUAL. The Path-finding algorithm can be used in conjunction with the feasibility condition [Garcia-Luna-Aceves and Murthy, 1997], which has been demonstrated to be more efficient than DUAL. A more promising approach is to combine the use of sequence number and the feasibility condition [Garcia-Luna-Aceves et al., 2003].

2.3.5 Summary

There is no silver bullet to address the count-to-infinity problem. Imposing a maximum value on hop count means that packets have to bounce within a loop for the maximum hops. Setting the limits too small will however restrict the length of paths. Split-horizon and poison reverse technique can only remove loops between neighbours. Path-finding algorithms make use of topology information to prevent loops. However, temporal loops may still occur and it is challenging to implement in an on-demand manner. The destination sequence number and the feasibility condition guarantee loop-freedom at every instant. The use of the sequence number is conservative. Every time a path breaks, a node requests for an increased sequence number of affected destinations, which invalid alternative paths with current sequence number. Using historical low value of distance, feasibility condition is also conservative and may lead to large amount of update messages among multiple nodes. In the end, the most promising approach for a specific context may be a combination of these techniques.

It should be noted that the term 'distance' we use in this subsection can be measured by but not limited to hop count. There can be many other routing metrics. We will review the use of different routing metrics in next subsection. The motivations and results of approaches that prevent loops in single path routing have a large influence on multipath routing and optimal routing, which we will discuss in subsection 2.1.5

Method	Loops	Drawback
Hop count	Temporal	Long delay before detected
Split-horizon with poison reverse	Yes	Prevent only one hop loops
Sequence number	No	Too Sensitive
Path-finding	Temporal	Need to maintain topology information
Feasibility condition	No	Sensitive

Table 2.1: Summary of Methods addressing the Count-to-Infinity Problems

2.4 Routing Metrics

In order to avoid potential confusion, we first distinguish routing objectives from routing metrics. A routing objective is a performance metric that a routing algorithm aim to optimize. In this thesis, we take delay of all traffic as our routing objective. On the other hand, a routing metric is a type of measurement to describe a path. Throughout this thesis, we interchangeably use the term 'distance', 'length' and 'cost' to denote the measurement of a path based on which a routing algorithm makes route selections.

Routing metrics play an important role in the behaviour of routing algorithms. Various types of metrics have been proposed to date. We focus our attentions to the most common type - additive metrics. Under an additive metric, the distance of a path is the summation all links of the path. We refer further interests for different types of metrics to corresponding results [Sobrinho, 2003, Baumann et al., 2006].

If hop count is used as the metric, the cost of every link is constantly one. Routing algorithms based on hop count are invariant to traffic dynamics of a network. Some routing protocols, such as OSPF and IS-IS, allow a customized metric. Network administrators can specify cost of each link based on estimated traffic and link capacity. In fact, this is an offline traffic engineering approach to apply optimal routing solution, which we will study in section 3. However, once specified, link weight of these protocols remains static during runtime.

Traffic aware metrics allow routing protocols to adapt to traffic dynamics. End-to-end delay is a traffic aware metric that measures the total delay of packets from source to destination. Routing algorithms based on end-to-end delay can avoid congested nodes and reduce the overall delay of a network. However, delay is a very sensitive metric. The cost of a path may change drastically to each routing decision, which causes fluctuation of routing solutions. The instability of routing algorithms based on a sensitive metric, such as delay, has been well observed and studied [Bertsekas, 1982, Wang and Crowcroft, 1992]. Many heuristic algorithms that take probabilistic routing decisions based on path cost have been developed. The System and Traffic dependent Adaptive Routing Algorithm (STARA) [Gupta and Kumar, 1997], as well as its successors [Borkar and Kumar, 2003, Raghunathan and Kumar, 2009], applies exponential moving averages onto paths cost and routing decisions to avoid sudden fluctuations. Another heuristic algorithm, Replex [Fischer et al., 2006], proposes a (α-β)-exploration-replication policy to stabilize routing solutions. These heuristic algorithms, although proposed to use delay as routing metrics, can be used in optimal routing algorithms based on gradient.

Many traffic aware metrics that are less sensitive have been proposed. They help to reduce the network delay without leading to instability. Expected transmission time (ETT) [Draves et al., 2004] uses MAC layer transmission time along a path as routing metric. Expected transmission count (ETX) [Couto et al., 2003] uses the number of MAC layer transmission and retransmission as an indicator of link quality. Metric of interference and channel switching (MIC) [Yang et al., 2005] considers intra- and inter-flow interference. It is also possible to model the routing problem through reinforcement learning approaches. Q-routing [Littman and Boyan, 1993] is a Q-learning algorithm. SAMPLE [Dowling et al., 2005] combines collaborative reinforcement learning technique. In these cases, the routing metrics is essentially the inverse of action rewards. Queueing delay accounts for a large portion of communication delay, especially in a congested scenario. Therefore, using queue length of forwarding nodes as routing metric can achieve good performance in term of delay [Basu et al., 2003].

2.4.1 Morality of Routing Algorithms

We refer to a routing algorithm to be selfish if each communication flow tries to optimize its own performance. Now that we have a better understanding towards routing metrics, we may define selfish routing from a different angle: *A routing algorithm is selfish if its routing metric is the same as it objective metric.* For example, a routing algorithm using end-to-end delay as routing metric is selfish for a delay minimization routing problem.

Indeed, describing a routing algorithm as selfish may be inaccurate, because routing algorithms do not have "attitudes" - they simply try to make the best routing choices based on a certain routing metric. Therefore, a fundamental question of the optimization problem is "what is the routing metric that presents the benefits of an entire network rather than individual communication, i.e. optimizes a given objective?"

A quick answer to this question is the marginal cost. A routing algorithm that uses marginal delay as routing metric can optimize the delay objective. One of our proposals is the derivation of a routing metric that captures the wireless marginal cost, which we will present in Chapter 3.

2.5 Multipath Routing

Multipath routing is a type of routing algorithms that discovers and uses multiple paths for each communication between a source and a destination pair. In comparison to single path routing, it makes use of more network resources to achieve better fault-tolerance, increased throughput, and reduced delay. We are interested in multipath routing algorithms because an optimal routing solution that minimizes delay of a network may use multiple paths. However, the majority of existing optimal routing algorithms do not address route discovery and establishment but simply assume the availability of all potential paths. In this subsection, we first describe functionalities of multipath routing. Then, we review three major types of multipath routing approaches. We argue that these approaches can not be applied to optimal routing algorithms for route discovery. Finally,

we identify the core issues to be addressed in order to design a multipath route discovery for optimal routing algorithms.

The task of multipath routing is more complex in comparison to single path routing. Its functionality can be divided into four interconnected components: message exchange mechanism, path filtering in route discovery, path selection after discovery, and load distribution. In the following, we will give an overview of these components to demonstrate how multipath routing works.

1) Control message exchange: This mechanism addresses how topology or route information is shared among network nodes in route discovery. Similar to its counterpart in single path routing, control message exchange in multipath routing is carried out primarily by either link state algorithms or distance vector algorithms.

Link state algorithms requires little modification to work in multipath cases. Network topology information retrieved at each node is sufficient to compute multiple paths. One example is the equal-cost multi-path (ECMP) routing, where multiple shortest paths instead of one are computed. ECMP is supported by both OSPF and IS-IS with no additional requirement on message exchanges.

Distance vector algorithms for multipath routing work with a modified distributed Bellman-Ford algorithm. During the propagation of path information from destination nodes to source nodes, intermediate nodes do not choose the single best distance but forward multiple distances back to source nodes. The criterion of selecting those multiple distances is explained in the next component. In the light of reducing communication overhead, distance information of multiple paths may be fused into one value at intermediate nodes. In this case, rules of the merge should be addressed.

In addition to link state routing and distance vector routing, link reversal algorithm [Gafni and Bertsekas, 1981] presents an alternative message exchange methods.

2) Path filtering during route discovery: There may be a large amount of potential paths between each source and destination pair, particularly in wireless ad hoc networks. It is both expensive and unnecessary to discover all of them. There can be

certain criteria used in the process of route discovery to filter out undesirable paths - depending on the angle, some refer them as rules to update desirable paths. Some of the notable criteria include loop-freedom, maximum length of paths, and disjoint paths. Loop-freedom is a common path filtering rule. Techniques that we have reviewed in section 2.3 can be extended to enforce this rule.

A popular design choice of filtering criterion is on the path disjointness. Following this criterion, only disjoint paths are discovered, which experience little correlation. Congestions and transmission failure of one path is less likely to affect its disjoint paths.

Path filtering can also be based on the distance - or the cost - of paths. For example, an upper bound of path distance can be set so that long paths are left out. Although such a filtering may seem simple, a bad upper bound choice may harm the performance of routing algorithm. It requires sophisticated design to choose a proper bound on path costs. In fact, our proposed solution includes a result in this regard.

We should point out that not all route discovery algorithms filter paths in route discovery, although filtering is desirable. Examples include proactive link state algorithms and certain source routing algorithm [Lee and Gerla, 2001].

The result of the route discovery process is a subset $\tilde{\mathcal{P}}_f$ of the set of all potential paths $\mathcal{P}$. The goal of the path filtering process can be summarized as to reduce the size of $\tilde{\mathcal{P}}_f$ as much as possible while retaining the desired performance of the data transmission.

3) Path selection after route discovery: Path selection may follow the same criteria as path filtering and is performed after the route discovery process. Path filtering reduces the overhead incurred by route discovery as it eliminates bad paths early. However, without a comparison between discovered paths, such a filtering may not be sufficient to remove all redundant paths. Once the route discovery process is done, information of paths in the set $\tilde{\mathcal{P}}_f$ are known to network nodes. With this knowledge, nodes may select a subset $\tilde{\mathcal{P}}_s$ for data transmission. In the case of distributed hop-by-hop routing, path selection can happen at both sources and intermediate nodes, returning a list of next hop to forward data to.

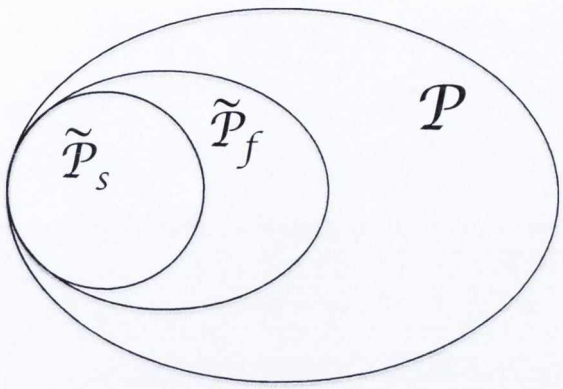


Fig. 2.2: Path Filtering and Selection

4) Load distribution: The final task of multipath routing is the data transmission, i.e. the distribution of data over multiple selected paths. Load distribution can be viewed as a function that takes costs of multiple paths as input. Such a function may have different forms. Single path forwarding algorithms distributes all data traffic to the single best paths - primary paths - while keep the rest of selected as backups in case that a primary path fails. ECMP [Moy, 1998] distributes data traffic of a flow evenly to multiple shortest paths. The Boltzmann distribution, also known as exponential penalty approach, have seen applications in both learning-based routing [Dowling et al., 2005] and traffic engineering-based routing [Xu et al., 2007]. A more advance approach is to distribute data load so that costs of all selected paths are equal. Selfish Wardrop routing [Gupta and Kumar, 1997] and optimal routing Wardrop routing [Yang and Weber, 2011] equalize the delay and the marginal delay of used paths respectively. Adaptive proportional routing [Nelakuditi and Zhang, 2002] splits data so that the blocking probabilities of multiple paths are equal. Some recent results [Kvalbein et al., 2009] separate load distribution metrics from the route discovery metrics, in order to avoid fluctuation of routes but stable yet responsive traffic distribution. [Kandula et al., 2005] follows a similar idea. In addition it applies a technique of congestion avoidance - XCP - on top of the heuristic convergence algorithm. to prevent oscillations. It is worthy noting that

the implementation of traffic splitting, although not our focus, is one active research field [Vutukury and Garcia-Luna-Aceves, 2000b].

Because our goal of this thesis is the design of an optimal routing algorithm that minimizes costs of network, we shall pay a special attention to the handle of path costs during each of the four processes. The route discovery process, in addition to establishing paths, should provide costs of paths. In the case of distributed hop-by-hop routing, merging of costs from multiple next hops should be addressed. For path filtering and selection processes, costs of paths can be used as a criterion. Even for criteria that are not based on directly on costs, it is desirable that they discover and select low cost paths. Finally, load distribution is dependent on and, if adaptive cost metrics are used, has an impact on costs of paths. Indeed, the handle of path costs acts as a fundamental connection among the four components of multipath routing.

If we view optimal routing as a type of multipath routing, the majority of optimal routing algorithms focus on load distribution component. Therefore, we are less interested in load distribution components in the domain of classical multipath routing algorithms but rather the rest of processes. It remains a question whether existing multipath routing algorithms can provide the functionality of the first three components that suits optimal routing. In the following, we will dive into existing multipath routing algorithms to examine their handle of paths cost.

2.5.1 Disjoint Routing in Ad Hoc Networks

It has been argued that correlations between the performance of different paths should be kept as small as possible. In this way, data transmission over multiple paths may be more resilient and reach a better delay and throughput performance. Disjoint routing methods provides multiple paths that do not intersect. Because of the high connectivity caused by wireless transmission, disjoint routing has been a popular design option for ad hoc networks.

Two paths are node-disjoint if they do not share any intermediate node. Split Multipath Routing (SMR) [Lee and Gerla, 2001] is based on Dynamic Source Routing (DSR) [Johnson et al., 2007] where a list of intermediate nodes of each path is kept at its source node. Two paths, the shortest path and its maximal disjoint path, are selected at each source node to carry data simultaneously. Based on AODV, AODV-Multipath (AODVM) [Ye et al., 2003] applies node-disjointness criterion in path filtering process. During route discovery, RREQ messages, including duplicate ones, are not filtered. Destination nodes response all RREQ messages with a RREP message. But rather, each intermediate node forwards only one RREP message through the shortest path to the source node. In this way, each intermediate node has at most one uplink towards the source node. After overhearing the RREP message, neighbours will not send other RREP messages to this node. Therefore, each intermediate node has at most one downlink towards the destination node. Node-disjointness is achieved.

The criterion of link-disjointness states that any two paths of one communication do not have common links. Ad hoc On-demand Multipath Distance Vector routing algorithm (AOMDV) [Marina and Das, 2002] makes an observation on the equivalent condition of link-disjointness: "every node on a path ensures that all paths to the destination from that node differ in their next and last hops". In order to apply this condition, they extend AODV to include both last hops and next hops in routing table for every path. RREQ or RREP messages are discarded if they come from the same last hop.

Node- and link-disjointness are not sufficient to eliminate correlations between two paths as neighbouring nodes or links may interfere with each other. In order to reduce the coupling among different paths of a communication, a stronger criterion called zone-disjointness has been proposed. AODVM with Path Diversity (AODVM/PD) [Mueller and Ghosal, 2005] keeps a value called *correlation factor* at each hop of a path, which is defined as the number of overheard RREP messages associated with the same route discovery. In addition to AODVM's filtering of node-joint paths, paths with a correlation factor higher than a threshold are filtered out at each nodes. Complete zone-disjoint

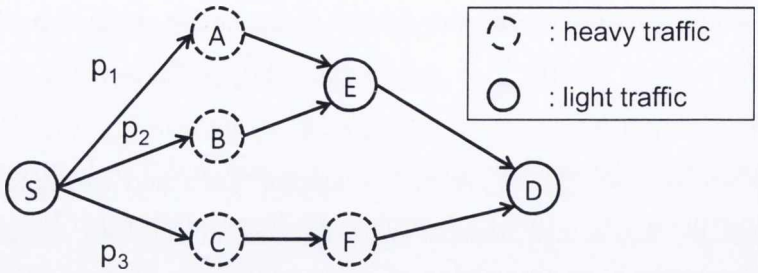


Fig. 2.3: Suboptimal Disjointness: P_1 and P_2 should be selected rather than P_3

paths where paths are out of each other's transmission range do not exist in general, but may be discovered with the help of directional antenna [Roy et al., 2003].

Other types of filtering and selection criteria can be applied in conjunction with disjointness. For example, loop-freedom is addressed in all aforementioned disjoint approaches, commonly using the destination sequence number mechanism. Another notable approach, Disjoint Path Selection Protocol (DPSP) [Papadimitratos et al., 2002] selects a set of link-disjoint paths only if statistics suggest they are reliable.

In disjoint routing methods, source nodes can get exact costs of discovered paths. The cost merge of multiple paths at intermediate nodes need not be addressed as paths do not intersect. Route disjointness is a strong condition. A large amount of potential paths will be ruled out, therefore overhead is not a problem. However, in wireless ad hoc networks, there may not exist multiple disjoint paths reducing the solution to a single path routing. What prevents disjoint routing methods being used for optimal routing algorithms is that the criterion of path disjointness does not result in optimal set of paths. As shown in Fig. 2.3, any type of disjoint routing algorithm will select one path from P_1 and P_2 and select path P_3. However, if P_3 is highly congested from other communication flows and if the bottleneck of P_1 and P_2 is not at the intersection node E, transmission over P_1 and P_2 simultaneously will result in better performance than disjoint routing.

In fact, this suboptimal results have been already realised within the field of disjoint

routing [Ye et al., 2003], where reliable nodes, such as node E in the example, are identified to allow path intersections. However, the identification of the reliable nodes lacks of theoretical discussion and justification.

2.5.2 Feasibility Condition-Based Methods

As we have established previously, if each node establishes a shortest distance towards a destination node, an order of nodes can be formed. As long as data forwarding follows a decreasing order of distances, loops will not happen. It means that at each hop, not only the best next hop but all neighbour nodes with a shorter distance can be selected without causing loops. In order to guarantee the correct order of nodes with the presence of topology changes, an extended feasibility condition can be applied in multipath routing.

Recall the notation in section 2.3.4, where D_j^i is the distance from node i to node j, D_{jn}^i is the copy of D_j^n kept at node i, and FD_j^i is the feasible distance from i to j. Denote the set of next hops selected for destination j at each node i as S_j^i. A feasibility condition for multipath routing, termed as loop-free invariant (LFI) condition initially, states [Vutukury and Garcia-Luna-Aceves, 1999]:

"Any routing algorithm designed such that the following two equations are always satisfied automatically provides loop-free paths at every instant, regardless of the type of routing algorithm being used: 1) $FD_j^i \leq D_{ji}^k$, $k \in N^i$; *2)* $S_j^i = \{k | D_{jk}^i < FD_j^i \wedge k \in N^i\}$"

The first rule comes from the definition of feasible distance FD_j^i, which is the historical low value of D_j^i. The second part is an extension of the second single path feasibility condition to the case of multiple next hops, which establishes the descending order through feasible distances. Note that the first rule of the single path feasibility condition is missing in the multipath counterpart. This because it is implied in the definition of D_j^i. Another difference between the single path condition and the multipath condition is that the former is proposed for distance vector routing while the latter suits for all types of algorithms.

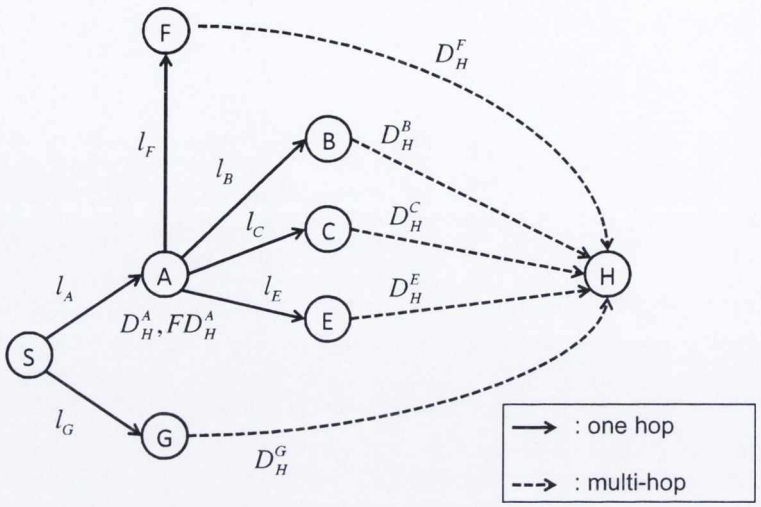


Fig. 2.4: A Example of Multipath Routing based on the Feasibility Condition

Notable methods includes DASM [Zaumen and Luna-Aceves, 1998], which is based on DUAL and uses diffusing computation to update topology changes; MPDA [Vutukury and Garcia-Luna-Aceves, 1999] is a link state algorithm. MPATH [Vutukury and Garcia-Luna-Aceves, 2000a] uses the second last hop information of the path-finding algorithm to construct network topology and applies Dijkstra algorithm, which is slightly different than the distance vector path-finding algorithm LPA. MDVA [Vutukury and Garcia-Luna-Aceves, 2001] uses a distance vector algorithm and is shown to outperform the previous methods.

These algorithms differ in their control message exchange mechanisms. Once a shortest path is established for every node to a destination node, the same feasibility condition is used to select next hops. We can illustrate this application through the behaviour of intermediate node A in the communication between S and H, shown in Fig. 2.4. Node A maintains a distance D_H^A and a feasible distance FD_H^A. It compares FD_H^A with reported distances from all its neighbours. Assuming only D_H^F and D_H^S are greater than or equal to FD_H^A, node B, C, E are selected as next hops. On receiving data from S, node A distributes the traffic over B, C and E according to their distances.

One advantage of multipath methods using the feasibility condition is that loop-freedom is guaranteed at all time. The merge of paths at intermediate node prevents large amount of information being propagated. For example, information about node B, C and E are hidden beyond node A. Each node makes decisions on load distribution over next hops locally - a desirable distributed manner.

However, path selection and load distribution at each intermediate node are based on reported distances of all neighbours. Therefore, existing multipath methods are all proactive algorithms, which introduces overhead.

More importantly, the merge of costs from multiple paths chooses the shortest distance, which may be inaccurate for certain cost metrics. We can demonstrate this problem by assuming the metric in Fig. 2.4 is delay. We assign the distance through node B as $l_B + D_H^B = 10$. Similarly, let the distance through node C and E be 15 and 15. Node F is not selected according to the feasibility condition. Choosing simply the smallest value, i.e. reported value from node B, D_H^A is set as 10. This value is accurate only if traffic is not split after node A and is otherwise inaccurate.

Now, imagine three units of data arrive at node A. Expected delay is then 3*10=30 (unit size*unit delay). However, what actually happens at A is that the 3 units are distributed and transmitted over three next hops simultaneously causing an overall delay of 15 (unit size*unit delay), which is 5 (unit delay) on average. In fact, one can easily observe that the actual merged cost of A is always less than the shorted distance if traffic is split between A and H. This underestimated cost of node A results in a suboptimal shift of load distribution at node S towards single path through node G.

2.5.3 Link Reversal Algorithm

For each destination node d, the link reversal algorithm [Gafni and Bertsekas, 1981] builds a directed acyclic graph (DAG) rooted at the destination, i.e. a loop-less graph where all links pointing to the direction of node d. The link reversal algorithm provides multiple alternative paths without considering their costs.

Gafni and Bertsekas considered a directed network without partition in the case of one destination. Two types of algorithms have been proposed: the full link reversal algorithm and the link partial reversal algorithms. Firstly, we describe the full link reversal algorithm. Initially, there are misdirected links in the network. Other than the destination, nodes without outgoing links are called sinks. At each round, sinks reverse the direction of all their incoming links. It has been shown that after at most b^2 rounds, where b is the initial number of misdirected links, no sinks exist in the network and all links are properly directed to form the DAG [Busch et al., 2003].

In the partial link reversal algorithm, sinks do not reverse all but some of their incoming links. It can be explained through a numbering scheme. A triple value $h_u := (\alpha_u, \beta_u, u)$ called height is associated with each node u at all time. α_u is essentially a sequence number, which initially is 0 for every node. β_u is some integer number. h_u is lexicographically ordered[1]. Similar to what we have reviewed in section 2.3.4,

Each non-destination node u maintains a list of its neighbours v that have previously reversed the link (u, v) to (v, u). At each round, each sink node u sets its height h_u as follows, which results in some of its coming links being reversed:

1. If the list includes all its neighbours, reverse all the incoming links.

2. Otherwise, set $\alpha_u = min_{w \in N_u}\{\alpha_w\} + 1$; For any neighbour v whose α_v equals the newly assigned α_u, set β_u to be smaller than β_v for all such neighbours. Then empty its list.

Through induction, it can shown that all neighbours in the list of u have the value of $min_{w \in N_u}\{\alpha_w\} + 1$. That means, in the case the list is not full, sink nodes reverse their incoming links that are not in their list. The most notable link reversal algorithm is Temporally Ordered Routing Algorithm(TORA) [Park and Corson, 1997]. TORA introduces an on-demand partial reversal algorithm with the ability to handle network

[1] two triples are lexicographically ordered $h_u > h_v$ if and only if one of the following is true: 1) $\alpha_u > \alpha_v$ 2) $\alpha_u = \alpha_v$ and $\beta_u > \beta_v$ 3) $\alpha_u = \alpha_v$ and $\beta_u = \beta_v$ and $u > v$

partitions. TORA uses a synchronized system time of each node as α. For each node u, TORA set α_u as the time of link failure based on a synchronized network system time. β_u is initialized with the shortest hop count from u to destination d.

Before any topology changes, the link reversal algorithm discovers all potential paths. Although it is essentially a variation of distance value, the height triple is not used for route discovery or load distribution, but rather to remove loops. Routing optimality is not a concern for the link reversal algorithm.

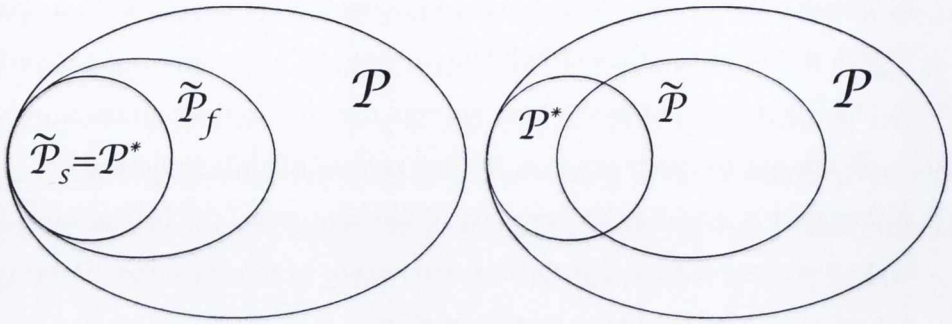
2.5.4 Summary

Optimal routing can be viewed as one type of multipath routing. It differs with classical multipath routing algorithms in that it focuses on the load distribution component using marginal delay as cost metric. A question arises naturally:

Is there a multipath routing approach that provides the route discovery and the route establishment functionalities to facilitate optimal routing algorithms?

The requirements for a multipath routing algorithm to work with an optimal routing approach include three parts. Firstly, it should prevent redundant path information from spreading in order to avoid excessive communicational and computational overhead. This can be done by the path filtering and selection processes: the smaller set of paths presented to the load distribution process, i.e. $\tilde{\mathcal{P}}_s \subset \tilde{\mathcal{P}}_f \subset \mathcal{P}$, the less overhead is incurred by route discovery. In addition, merging information from different paths when they meet alleviates overhead.

Secondly, the resulting path set of multipath route discovery and selection should include the optimal path set. Given a network topology and traffic demands in the network, there exists an optimal set of paths, denoted as $\mathcal{P}^*$, over which an optimal routing solution distributes non-zero traffic. A naive approach of multipath routing that meets this requirement is to flood RREQ messages and to discover all potential paths, although this creates large overhead. In conjunction with the first requirement, the ideal path filtering process should reduce the size of $\tilde{\mathcal{P}}_f$ as much as possible while containing



(a) Ideal Case (b) Undesirable Case

Fig. 2.5: Path Filtering and Selection in Relation to Optimal Path Set

the optimal set; The ideal selection of the path set $\tilde{\mathcal{P}}_s$ should coincide with the optimal set. These set relations are demonstrated in Fig. 2.5(a). However, it is challenging to design ideal criteria for filtering and selection processes. Under ill-considered criteria, some paths from the set $\mathcal{P}^*$ may not be discovered or selected. This undesirable result can be formally written as $\mathcal{P}^* \not\subset \tilde{\mathcal{P}}_s \subset \tilde{\mathcal{P}}_f$, which is shown in Fig. 2.5(b).

Thirdly, the exact costs of paths should be provided to nodes who make load distribution decisions. A cost of a path reported from one node, say A, to the decision-making node, say S, is exact - or accurate - if and only if one of the following is true: 1) it is a single path from node A to its destination node 2) if it consists of multiple paths merged at certain intermediate node, the merge of paths does not affect the traffic assignment from node S to node A.

A comparison of the three types of algorithms with respect to the three requirements is shown Fig. 2.6. Disjoint routing algorithms filter a large amount of paths thus overhead is reduced. Because paths do not merge, exact costs of paths are available. However, the criterion of route disjointness can not guarantee the optimal path set. Feasibility condition-based methods filters only paths with risk of loops, thus the optimal path set is likely to be provided. Costs of multiple paths are merged at intermediate nodes, therefore large overhead is prevented from spreading. However, only a proac-

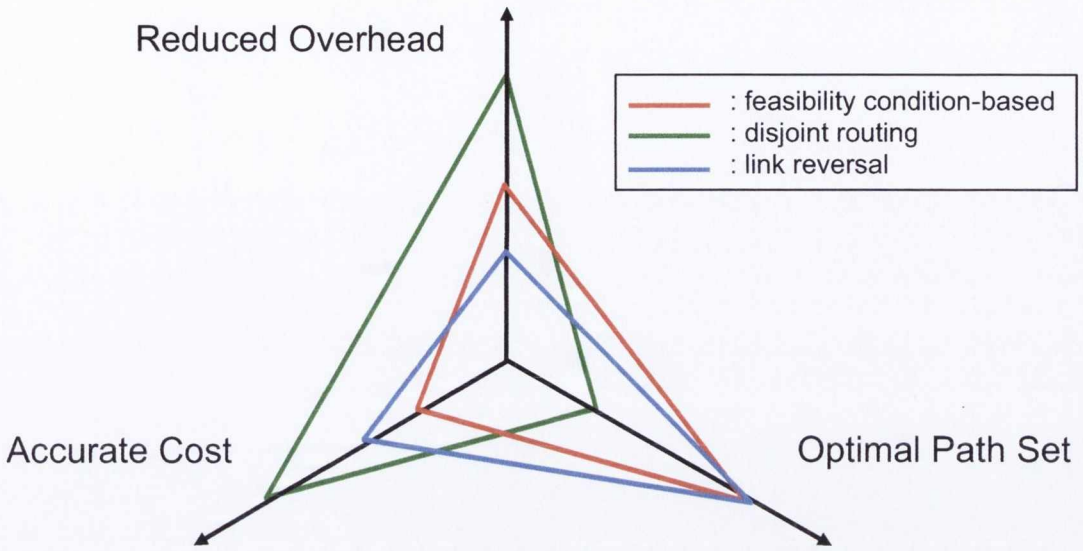


Fig. 2.6: Evaluation of Multipath Routing: The three axes represent three requirements. A solution that meets all three requirements is missing!

tive manner is supported in feasibility condition-based methods, which introduces extra overheads. The link reversal routing algorithm discovers all loop-free paths regardless of their costs. Optimal path set is provided at source nodes. Exact cost can potentially be provided, but is not properly addressed. Because paths are weakly filtered with loop-freedom criterion, there may be a large amount of potential paths presented at source nodes, which creates large overhead.

In summary, existing multipath routing algorithms in the literature do not suit the requirements of optimal routing.

2.5.5 Core Issues of Multipath Routing

In last subsection, we have summarised the missing bit of multipath routing to support optimal routing. Now we can take one step further to examine core issues to be addressed for potential solutions.

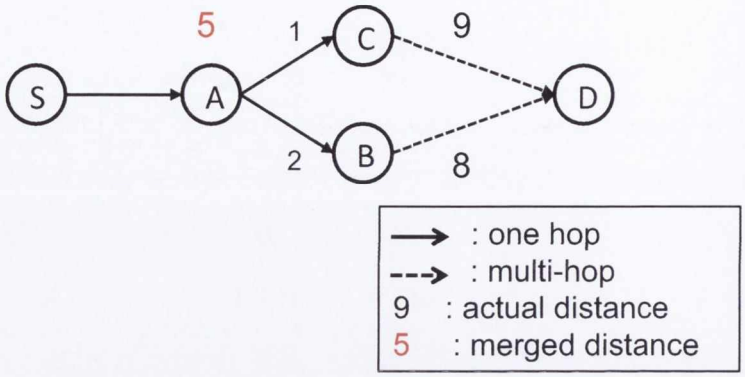


Fig. 2.7: Conflict between exact merged costs and loop-freedom

1) Path filtering and selection criteria should be based on path costs. Optimal routing algorithms minimize the delay experienced by all traffic. Roughly speaking, paths that are eliminated during route discovery should have a large cost and paths that are in $\mathcal{P}^*$ should have have a small cost. Therefore, criteria based on path costs could meet both the first and the second requirements aforementioned. However, it remains an open issue to choose a proper value of path costs as a criterion. Such a choice differs among each communication of the same network. It depends on the traffic demand of one flow as well as runtime network traffic, both may vary over time.

2) For certain cost metrics, such as marginal delay, it is a hard task to merge costs of multiple paths. One natural solution is to find the function of delay for merged paths and to apply differentiation. However, the exact form of the function may not be possible and the function may not be differentiable. A new perspective is needed to compute merged costs.

It turns out that even if exact merged costs can be computed, the value may be smaller than individual costs, which conflicts with the criterion of loop-freedom. Consider an example with Fig. 2.7, the merged cost of node A is 5 - smaller than both 8 and 9. Since the distance is no longer strictly decreasing to destination D, traffic from S may travel back from node C and B to node A. In the author's opinion, this may contribute to the choice of minimum value for merged costs in feasibility condition-based methods.

Name	Triggering	Message Exchange	Cost Merge of Multiple Paths	Path Filtering	Path Selection	Load Dist
SMR	on-demand	source routing	no	no	shortest path & its maximal node-disjoint path	primary path
AODVM	on-demand	distance vector	no	node-disjoint paths	n/a	n/a
AODVM/PD	on-demand	distance vector	no	node-disjoint paths	maximal zone disjoint paths	n/a
AOMDV	on-demand	distance vector	no	link-disjoint paths	n/a	n/a
DSDP	n/a	n/a	no	no	link-disjoint and reliable paths	n/a
DASM	proactive	diffusing computation	yes: minimum distance	no	feasibility condition	n/a
MPDA	proactive	link state	yes: minimum distance	no	feasibility condition	local equilibrium
MPATH	proactive	path-finding	yes: minimum distance	no	feasibility condition	n/a
MDVA	proactive	link state	yes: minimum distance	no	feasibility condition	n/a
TORA	on-demand	link reversal	no	no	n/a	n/a
OSPF-ECMP	proactive	link state	no	no	equal cost paths	equal share
STARA	n/a	flooding	no	no	no	equilibrium

Table 2.2: Summary of Multipath Routing Protocols Reviewed

Chapter 3

Optimal Routing Approaches

The review of existing multipath routing algorithms in the previous section provides us the techniques as well as the challenges to discover multiple paths for optimal routing algorithms. With all paths established, a routing decision of a node determines a load distribution over multiple paths. Routing decisions of all nodes correspond to a traffic pattern of the network. Under a certain traffic demand, an optimal routing algorithm computes the set of routing decisions whose traffic pattern incurs the least overall cost of the network. In this section, we review the state of the art approaches of optimal routing.

In section 3.1, we give an overview of the mathematical background of optimal routing algorithms. In section 3.2 we examine different formulations of the routing optimization problem and their practical implications. In section 3.3 to 3.5, we discuss different types of optimization algorithms in detail.

3.1 Routing as an Optimization Task

Optimal routing algorithms are better suited to a dense network with heavy traffic demand. Interestingly, the first results of optimal routing can be found well before the prevalence of computer networks. In the domain of transportation control, the study

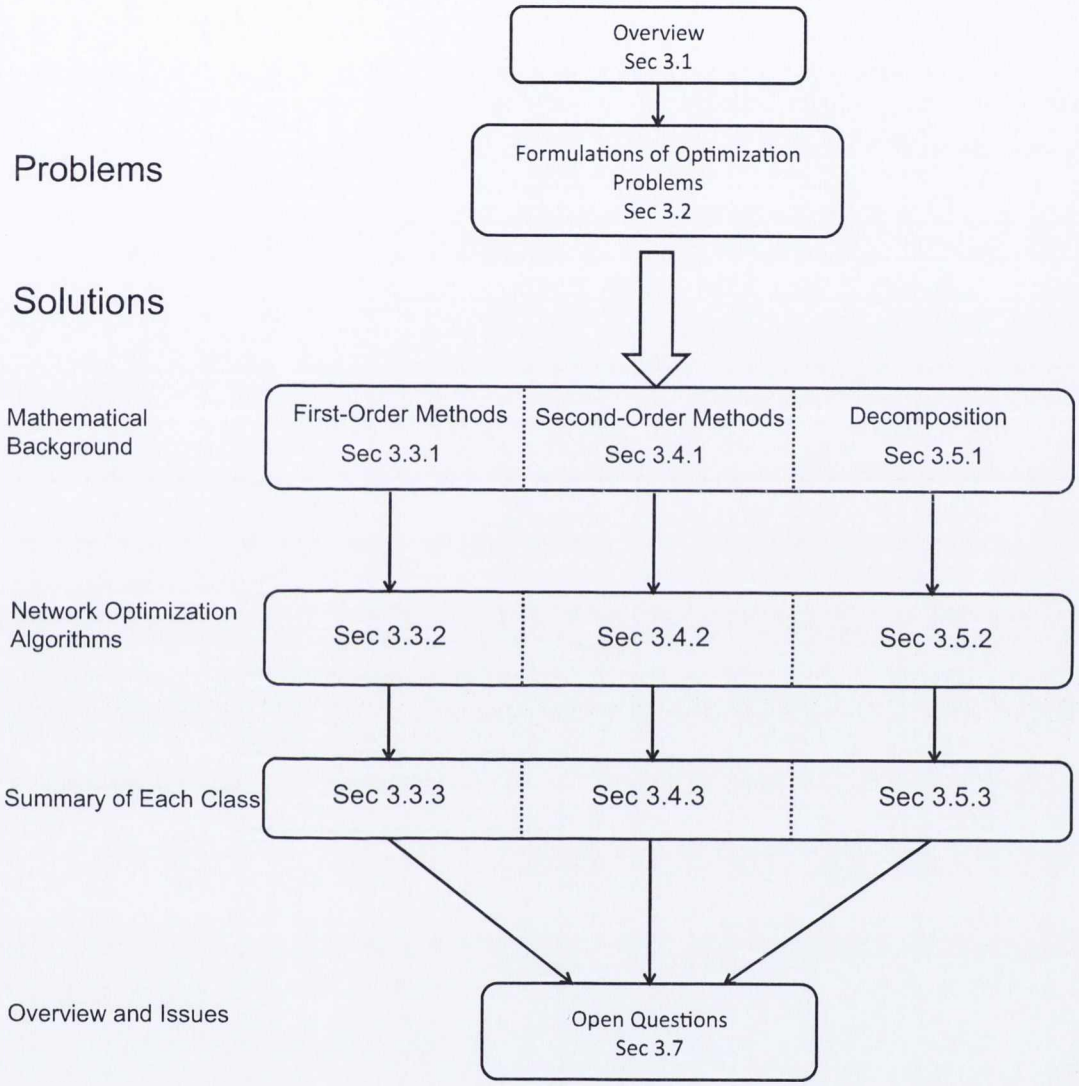


Fig. 3.1: Structure of Review on Optimal Approaches

of road traffic assignment and management dates back to 1950s. Wardrop equilibrium, which is essentially a Nash equilibrium of flows, has been proposed to model road traffic [Wardrop, 1952]. It was later demonstrated that a Wardrop equilibrium under marginal cost coincides with the system optimum [Beckmann et al., 1956]. These results can be applied to the domain of telecommunications with only minor modifications.

Optimal routing approaches can be viewed as an application of the well-studied (convex) optimization theory [Boyd and Vandenberghe, 2004]. We start with an overview of the theoretical background before going to the details of different types of optimization algorithms. The problem addressed by optimization theory can be formulated as follows:

$$\text{minimize } y(x) \tag{3.1}$$

$$\text{subject to } g_i(x) \leq 0,\ i = 1...m \tag{3.2}$$

$$h_i(x) = 0,\ i = 1...cm \tag{3.3}$$

where $y, g_i, h_i : \mathbb{R}^n \to \mathbb{R}$. It states that the optimization problem is to find an x that minimizes the objective function $y(x)$, (3.1), among all x that meet the inequality constraints (3.2) and the equality constraints (3.3). This is the basic form of an optimization problem. Applying to the optimal routing domain, the objective function can be either an cost function to be minimized or a utility function to be maximized. The vector variable x may have different representations in different algorithms, such as source rate, link rate, and etc. The equality and inequality may have different forms and meanings in different algorithms, depending on their focuses. While the nature of the optimization problems remains the same, i.e. it can be summarized by the form of (3.1) - (3.3), these different formulations of the problem may have significant differences in their practical implications. We will discuss some of the common formulations in subsection 3.2.

For most cases in practise, such as optimal routing, the optimization problem can not be solved analytically but requires an iterative approach. Denote the optimal solution as y^*. An iterative algorithm start with a given initial value $x^{(0)}$ in the domain of the

objective function, denoted as $x^{(0)} \in \mathbf{dom}\ y$. A series of $\{x^{(t)}\} \subset \mathbf{dom}\ y$ is computed in the form of

$$x^{(t)} = x^{(t-1)} + \alpha * \Delta x^{(t-1)}$$

where $\Delta x^{(t-1)}$ is the *search direction* and α is real value called *step size*. A solution $x^{(t)}$ is *feasible* if it meets condition (3.3) and (3.2). An algorithm is *convergent* to optimum, which we refer to as an optimal algorithm, if

$$y(x^{(t)}) \to y^* \text{ as } t \to \infty$$

The algorithm stops if $x^{(t)}$ is feasible and that $y(x^{(t)})$ is within a small neighbourhood of the optimal value, i.e. $|y(x^{(t)}) - y^*| < \epsilon$ for some small positive value ϵ. We can translate the mathematics to the scenario of an optimal routing algorithm: Routing decisions at each iterative step of a network dictate a traffic pattern of the network. Based on costs incurred, the algorithm computes changes in routing decisions and distributes data traffic based on the new routing decisions. This completes one step of the convergence.

Algorithm 1 General Form of Iterative Optimization Methods

Starting with an initial value $x^{(0)}$ and $t = 0$, **repeat**:

1. $t := t + 1$
2. Compute the search direction $\Delta x^{(t)}$
3. Decide a step size α
4. Update solution $x^{(t+1)} = x^{(t)} + \Delta x^{(t)}$

unitl the stopping criterion is satisfied.

Many optimal routing algorithms have been proposed to compute the series of search steps. We are interested in two features of an given optimal routing algorithm : the convergence rate and the distributed manner.

3.1.1 Convergence Rate of Optimal Routing Algorithms

One important measurement of an optimal routing algorithm is the number of iterative steps the algorithm takes from a starting point to an optimal solution. In the context

of numerical analysis, the convergence rate of an iterative algorithm describes the speed at which its solution series $\{x^{(t)}\}$ approaches the optimal solution. Formally speaking, it is defined as

$$\mu = \lim_{t \to \infty} \frac{|y(x^{(t+1)}) - y^*|}{|y(x^{(t)}) - y^*|}$$

An optimal algorithm is said to converge linearly if $\mu \in (0, 1)$. If $\mu = 1$, then the convergence is superlinear; If $\mu = 0$, then the convergence is sublinear. The convergence rate of an algorithm is a major factor of the number of steps required to converge. Minor factors include the choice of the starting point and the stopping criterion ϵ. The convergence rate of an algorithm is primarily dependent on the orders[1] of underlying methods: The first order (sub)gradient-based algorithms generally show a linear or a sublinear convergence rate, while the second order Newton-based algorithms exhibit a much faster superlinear convergence rate. However, this concept of convergence rate from the domain of numerical analysis focuses on the computational aspect of an algorithm. It may not tell the complete truth about an optimal routing algorithm.

The delay overhead incurred by multihop communications, particularly in wireless ad hoc networks, can be excessive in comparison to the delay incurred by computation at one node. Computation for one step takes $O(n^3)$ arithmetic calculations at most, most complex step being the second-order Newton step. Consider an example where a source node wishes to distribute traffic over 1000 paths, which is more than being realistic. The time needed to compute one step with a CPU at a gigahertz (GHz) level of clock rate is at the scale of milliseconds. On the other hand, multihop information exchange may be invoked in order to evaluate the objective cost function $y(x)$ at each step. According to analytical results [Malone et al., 2007], the one-hop transmission delay for IEEE 802.11 (11Mbps) with non-saturated traffic load is at the scale of 10 milliseconds. Because the one-hop transmission delay is only a fraction of the multihop end-to-end delay, steps that involves multihop information exchange introduce large delay. It should not be treated

[1]The term "order" refers to the degree of Taylor approximation to the objective function. Bear with this concept for now, which will come clear once we dive into the mathematical background of each type of algorithms.

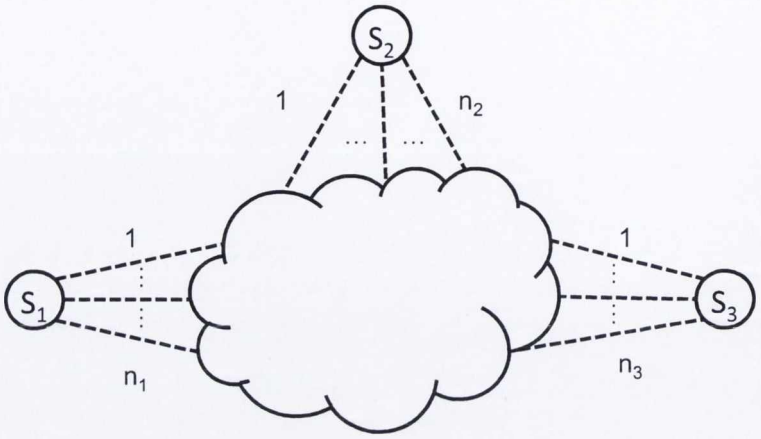
equally with steps that computed locally. Therefore, in addition to the measurement of convergence rate, we shall pay a special attention to the number of information exchanges incurred by the solution series $\{x^{(t)}\}$ of an optimal algorithm.

3.1.2 Distributed Manner of Optimal Routing Algorithms

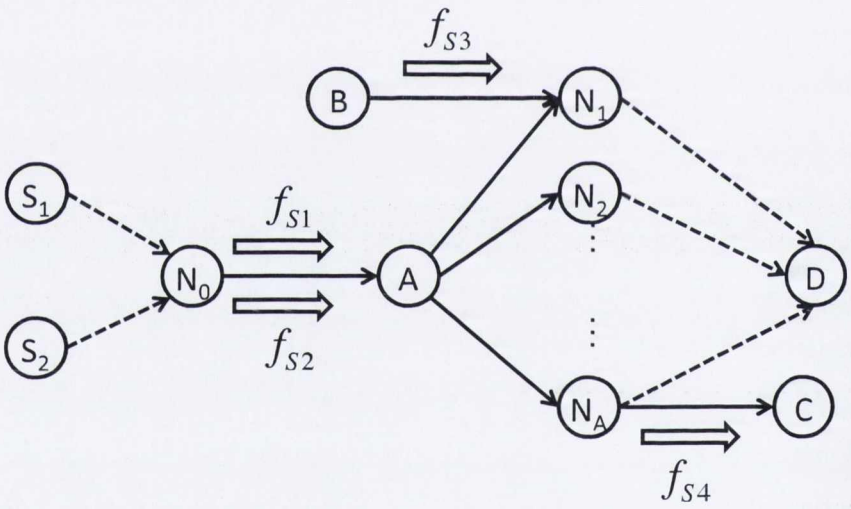
The distributed manner of an optimal routing algorithm can be classified into flow-distributed and node-distributed. A routing approach is flow-distributed if each communication flow, identified by a source-destination pair, computes its own routing decisions, as shown in Fig. 3.2(a). Unlike the case of classical (selfish) routing approaches, flow-distributed implementation is not straightforward for optimal routing. From the problem formulation, we note that different flows are coupled in two ways: 1) through the objective function (3.1). For example, the behaviour of one flow may affect the delay experienced by other flows; 2) through the inequality and equality constraints (3.2, 3.3). For example, the behaviour of one flow may affect the available capacity for other flows.

A routing algorithm is node-distributed if routing decisions are made locally at each intermediate node, as shown in Fig. 3.2(b). Optimal routing algorithms, especially the second order algorithms, aim to optimize the network traffic as a whole. The main challenge for node-distributed algorithms comes from the conflict between the global optimum and the limited view of each intermediate node. A simple approach to implement a node-distributed algorithm is to duplicate global knowledge of the entire network. However, this approach is undesirable as it creates large control overhead. The key to both flow- and node-distributed manners is to decompose the network optimization problem into subproblems that can be addressed separately at different nodes.

We organise our related work according to the orders of algorithms with a special attention to their distributed manners. In subsection 3.3, we review existing first order algorithms. In subsection3.4, we present existing second order Newton-based algorithms. In subsection 3.5, we discuss a common technique that decomposes Newton-based algorithms into distributed manner. After the decomposition, a part of the approach runs



(a) Flow-Distributed: Each source makes routing decisions over its multiple paths, taking its impacts on other traffic flows into considerations



(b) Node-Distributed: Each intermediate node makes routing decisions over multiple next hops for arriving traffic. The routing decision of f_{S1} at node A should cooperate with the routing of flow f_{S2}, f_{S3}, and f_{S4} at neighbours as well as the traffic distribution at source $S1$ and $S2$

Fig. 3.2: Distributed Manners of Optimal Routing Algorithms

first order algorithms and the other part of the approach runs second order algorithms. With a slight abuse of terminology, this method can be seen as mixed orders.

It is interesting that one may find optimal approaches in the domain of congestion control and flow control [Chiang et al., 2007] that share many similarities with optimal routing algorithms. In this review, we will clarify their differences in subsection 3.2, but otherwise treat their essential mathematical approaches as related works.

3.2 Variations of Network Optimization Formulations

We have formulated the optimal routing problem to be addressed as (1.2a). Now, let us temporally detach the routing implication from the model and focus on its mathematical presentation: The problem (1.2a) is a constrained convex optimization problem, which follows the general model (3.1). A vast body of network optimization problems, including congestion control, routing and link scheduling, can be seen as certain forms of the general model. Algorithms that solves these problems are, to a certain extend, interchangeable. For this reason, our state of the art review will include notable algorithms from different areas of network optimizations, such as congestion control. However, before our review of these algorithms, it is important to understand the differences and the similarities between various forms of the optimization model. In this section, we discuss different formulations of network optimization problems.

3.2.1 Modelling of Optimal Routing Problems

In the domain of optimal routing, we aim to find a traffic pattern $\mathcal{F} = \{f_p\}$, $p \in \mathcal{P}$ of the network that minimizes the overall cost, where $\mathcal{P}$ is the set of all potential paths of all communication flows. In order to differentiate from the general form (3.1), we write

the optimal routing problem as follows:

$$\text{minimize } C(\mathcal{F}) \tag{3.4}$$

$$\text{subject to } g_i(\mathcal{F}) \leq 0,\ i = 1...m \tag{3.5}$$

$$h_i(\mathcal{F}) = 0,\ i = 1...cm \tag{3.6}$$

A common choice of function g is $\mathcal{RF} - \mathcal{C}$, where $\mathcal{C} \in \mathbb{R}^{m\times 1}$ is link capacity vector and $\mathcal{R} \in \mathbb{R}^{m\times n}$ is the routing matrix indicating whether a link belongs to a path. $\mathcal{R}_{i,j} = 1$ is path j travels through link i, vice versa.

A common choice of function h is $A\mathcal{F} - \mathcal{T}$, where $A \in \mathbb{R}^{cm\times n}$ indicates paths that belong to the same communication. $\mathcal{T} \in \mathbb{R}^{cm\times 1}$ is the traffic demand vector for each communication.

3.2.2 Handle Variables and Constraints of Optimization Problems

It is sometimes sensible to translate the constraints from the form of equations to its equivalent form of a set. Define the *feasible domain* of the problem (3.1) as the set of all $\mathcal{F} \in$ **dom** C that satify constraints (3.5) and (3.6), denoted as $\mathcal{FD}$. The optimization problem is then to minimize (3.4) over the set $\mathcal{FD}$, written as:

$$\text{minimize } C(\mathcal{F})$$

$$\text{subject to: } \mathcal{F} \in \mathcal{FD}$$

The common assumption for this type of notation is that $\mathcal{FD}$ is a closed set. Strictly speaking, this does not always stand. However, in practice, this is a trivial assumption that can always be guaranteed. For example, consider a cost function that represents the queueing delay of a M/M/1 queue: $C(f) = \frac{1}{\text{Cap}-f}$, where Cap is the capacity of the queue. The inequality constraint is that the flow f is non-negative. According to the domain of cost function and the constraint, the feasible domain of the problem is the interval $[0, \text{Cap})$, which is an open set. In practise, however, we can manipulate $\mathcal{FD}$ into $[0, \text{Cap} - 0.000001]$. Then we are able to apply the gradient projection method on

this closed set. It can be demonstrated that the final solution of the gradient projection method can be infinitely close to, if not the exact, the optimal solution - as we add more zeros to the manipulation.

3.2.3 Utility Maximization v.s. Cost Minimization

A similar mathematical formulation of the optimal routing problem is the network utility maximization (NUM) problem.

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{i=S} U_i(\mathcal{T}_i) \\ \text{subject to} \quad & \mathcal{R}\mathcal{T} \leq \mathcal{C} \\ & \mathcal{T} \succ 0 \end{aligned} \tag{3.7}$$

where $U_i(\cdot)$ is concave function, for example a common choice is $U_i(\mathcal{T}_i) = log(\mathcal{T})$. Apparently, this formulation shares many similarity with the cost minimization formulation. Indeed, as we will see, many but not all of the algorithms for example [Zymnis et al., 2007] that are used to address one formulation may be modified trivially to address the other formulation.

Roughly speaking, a NUM problem aims to transmit as much data as possible within the capacity constraints – the utility to be maximized is an increasing function with regard to throughput. Since the utility function is concave, the maximization focuses more on improving throughput of traffic flows that are experiencing less throughput. Therefore, it also enforces a certain level of fairness.

As shown in the formulation, NUM problem usually implies single path communication, although some exception may be found with a slightly different formulation [Wang et al., 2003].

3.3 (Sub)Gradient-based Methods

In this section, we review a class of optimal routing algorithms that is based on first-order optimization methods. These algorithms dates back to 1970s [Cantor and Gerla, 1974, Gallager, 1977]. A major drawback of the first-order class is their slow convergence. However, both first-order optimization methods and routing algorithms based on them play an important role of stepping stone for the second-order methods. In the following, we firstly describe the mathematical background of first-order methods. Then, we discuss their applications to optimal routing algorithms.

3.3.1 A Mathematical Background of First-Order Methods

In the following, we first describe the gradient descent method for unconstrained optimization. Then, we explain why the gradient descent method is a first-order optimization method. Finally, we illustrate a technique called gradient projection method to address constraints in gradient descent algorithms.

3.3.1.1 Unconstrained Optimization

We start with the minimization of a convex continuous cost function $C(\cdot) : \mathbb{R}^n \to \mathbb{R}$ without any constraints. We wish to find a series of traffic pattern $\{\mathcal{F}^{(t)}\}$ that converges to the optimal solution. One natural method is the gradient descent method.

The gradient of the function $C(\mathcal{F})$, denoted as $\nabla C(\mathcal{F}) = \{\frac{\partial C(\mathcal{F})}{\partial f_1}, \cdots, \frac{\partial C(\mathcal{F})}{\partial f_n}\}^T$, is an $n \times 1$ column vector of partial derivatives of $C(\cdot)$. The direction of the vector $\nabla C(\mathcal{F})$ points to the steepest increase of the function and the magnitude of $\nabla C(\mathcal{F})$ indicates the rate of the steepest increases. Therefore at any iteration t, aiming to minimize the cost function, a straightforward search direction is the steepest decrease, i.e. $\Delta\mathcal{F}^{(t)} = -\nabla C(\mathcal{F}^{(t)})$. The gradient of a function is a local property at each point. For example as show in Fig. 3.3, gradients of of points along the direction of $-\nabla C(\mathcal{F}_b)$ change gradually. Therefore, we need to set a step size, denoted as α, to go along the direction $-\nabla C(\mathcal{F})$

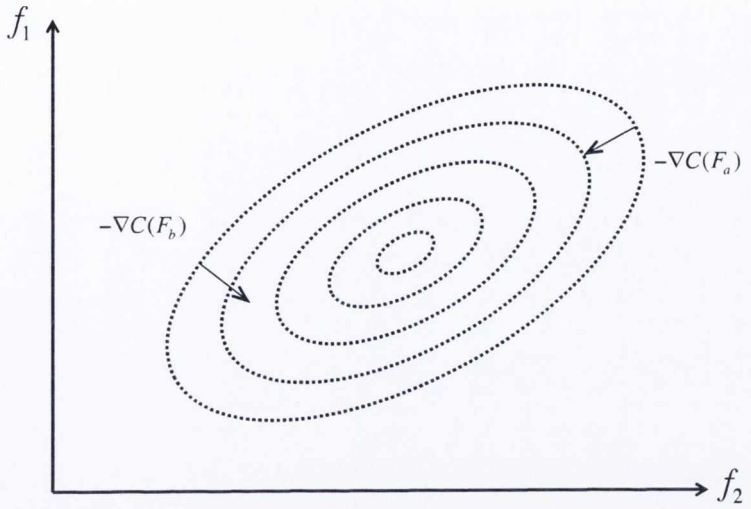


Fig. 3.3: A contour line demonstration of an optimization problem: Variables are two-dimensional, $\mathcal{F} = \{f_1, f_2\}$. The dashed contour line represents costs of the same value. $\mathcal{F}_a$ and $\mathcal{F}_b$ are two points. Their gradients are $\nabla\mathcal{F}_a$ and $\nabla\mathcal{F}_b$ respectively.

before compute again. The step size α can be assigned with a small constant value in the interval $0 < \alpha < 1$. With special care, we can also assign a diminishing value to α at each iteration. A more advanced approach is to compute α dynamically from cost information at each iteration step t, which achieves a better convergence rate. The computation of a dynamic α often requires an iterative method, such as the backtracking line search method [Boyd and Vandenberghe, 2004], that evaluates the objective cost function multiple times. It requires gathering global cost information, which as we have argued in 3.1.1 is a less desirable iteration. Some efforts can be seen in the distributed computation of the line search method [Zargham et al., 2012].

The iteration of gradient descent method can be summarized as follows:

$$\mathcal{F}^{(t+1)} = \mathcal{F}^{(t)} + \alpha\Delta\mathcal{F}^{(t)} = \mathcal{F}^{(t)} - \alpha\nabla C(\mathcal{F}^{(t)}) \tag{3.8}$$

Now we explain why the gradient descent method is a first-order optimization method. The gradient descent method belongs to a wider class of methods, called the steepest

descent method. Consider the first-order Taylor approximation of the objective cost function $C(\mathcal{F}^{(t+1)})$ at the $(t+1)$th iteration:

$$C(\mathcal{F}^{(t+1)}) = C(\mathcal{F}^{(t)} + \alpha\Delta\mathcal{F}^{(t)}) \approx C(\mathcal{F}^{(t)}) + \alpha\nabla C(\mathcal{F}^{(t)})^T\Delta\mathcal{F}^{(t)} \tag{3.9}$$

The motivation for the steepest descent method is to find a search step $\Delta\mathcal{F}^{(t)}$ so that the cost change per unit length of the search step is as negative as possible. That is to minimize $C(\mathcal{F}^{(t+1)}) - C(\mathcal{F}^{(t)}) \approx \alpha\nabla C(\mathcal{F}^{(t)})^T\Delta\mathcal{F}^{(t)}$. We normalize the search step with a unit length, denoted as $\Delta\mathcal{F}_{unit} = \frac{\Delta\mathcal{F}}{\|\Delta\mathcal{F}\|}$. The operator $\|\cdot\|$ represents any norm function, which returns a non-negative value representing the "length" of a vector. The steepest descent method can be described as:

$$\Delta\mathcal{F}^{(t)}_{unit} = \text{argmin}_{\mu}\{\nabla C(\mathcal{F}^{(t)})^T\mu \mid \|u\| = 1\}$$

If the norm function uses the Euclidean norm, the normalized step $\Delta\mathcal{F}^{(t)}_{unit} = -\frac{\nabla C(\mathcal{F}^{(t)})}{\|\nabla C(\mathcal{F}^{(t)})\|}$, whose corresponding unnormalized search direction $\Delta\mathcal{F}^{(t)} = -\nabla C(\mathcal{F}^{(t)})$. That is, the gradient descent method is a steepest descent method under Euclidean norm. While the Euclidean norm may be the most common choice, other norms can be used for certain problems.

One premise of the gradient-based method is that the cost function is differentiable. Indeed, a differentiable cost function is a widely adopted assumption by optimal routing algorithms to the best of our knowledge. However, this premise may not always hold. In certain cases, the problem we try to optimize may not be differentiable. We will for example encounter a non-differentiable problem if we decompose the original problem into a number of subproblems, which we will describe in subsection 3.5.1.3. In such a case, a subgradient method is needed.

A vector μ is a subgradient of the function $C(\cdot)$ at $\mathcal{F}_0$ if it meets:

$$C(\mathcal{F}) \geq C(\mathcal{F}_0) + \mu^T(\mathcal{F} - \mathcal{F}_0) \text{ for all } \mathcal{F}$$

For a convex differential function, the gradient of the function meets also the condition above. That is, the subgradient is a generalization of the gradient. If exists, the subgra-

dient may not be a unique value. The set of all subgradients, called the subdifferential and denoted as $\partial C(\mathcal{F}_0)$. The subgradient method works in a similar way as the gradient method, but with a different update formula than equation (3.8)

$$\mathcal{F}^{(t+1)} = \mathcal{F}^{(t)} - \alpha\mu, \quad \text{for any} \quad \mu \in \partial C(\mathcal{F}^{(t)}) \tag{3.10}$$

Unlike the gradient method, the subgradient method is not a type of steepest descent method. In fact, it is not a descent method in the sense that not every step results in a cost reduction. However, it can be shown that with a sufficient small step size α, the subgradient method is convergent.

3.3.1.2 Constrained Optimization

Now we consider the case where constraints are present in an optimization problem. We have seen mainly three types of techniques that can be applied to the gradient descent method. The first approach is the affine transformation to address equality constraints. The second approach is the projection gradient method for general equality and inequality constraints. The first two types of approaches can be used in conjunction in practise. The third approach is the Frank-Wolfe method for to linear equality and inequality constraints.

For an linear equality-constrained problem

$$\begin{aligned} &\text{minimize} \;\; C(\mathcal{F}) \\ &\text{subject to} \;\; A\mathcal{F} = b \end{aligned}$$

the equality constraints can be eliminated by rewriting the feasible set $\{\mathcal{F}|A\mathcal{F} = b\} = \{\tilde{A}\mathcal{Z} + \hat{\mathcal{F}}|\mathcal{Z} \in \mathbb{R}^{n-cm}\}$, where $\tilde{A}$ is a $n \times (n - cm)$ matrix whose column space is the null space of matrix A and $\hat{\mathcal{F}}$ is any solution of $A\mathcal{F} = b$. According to the definitions of column space and null space, $A(\tilde{A}\mathcal{Z}) = 0$ for any $\mathcal{Z} \in \mathbb{R}^{n-cm}$. Because $\hat{\mathcal{F}}$ is a feasible solution, we have $A\hat{\mathcal{F}} = b$. That means, for any $\mathcal{Z} \in \mathbb{R}^{n-cm}$, $\tilde{A}\mathcal{Z} + \hat{\mathcal{F}}$ automatically

meets the equality constraints

$$A(\tilde{A}\mathcal{Z} + \hat{\mathcal{F}}) = A(\tilde{A}\mathcal{Z}) + A\hat{\mathcal{F}} = 0 + b = b$$

Therefore, the linear equality-constrained constrained is equivalent to the following unconstrained problem:

$$\text{minimize } \tilde{C}(\mathcal{Z}) = C(\tilde{A}\mathcal{Z} + \hat{\mathcal{F}})$$

Finding the matrix $\tilde{A}$ and one solution $\hat{\mathcal{F}}$ to $A\mathcal{F} = b$ is in general not challenging. In particular, the equality constraints in an optimal routing problem may be easily addressed. For example, one common equality-constraint is

$$\sum_p f_p = \mathcal{T}_w \text{ for } p \in \mathcal{P}_w$$

Through affine transformation, we can transform the objective function as follows and eliminate the constraints:

$$\begin{aligned} &\text{minimize}_{f_1 \ldots f_n} \; C(f_1, ..., f_n) \\ &\text{subject to } \sum_{i=1}^{n} f_i = \mathcal{T} \\ &\Downarrow \\ &\text{minimize}_{f_1, \ldots, f_{n-1}} \; \tilde{C}(f_1, ..., f_{n-1}) = C(f_1, ..., f_{n-1}, \mathcal{T} - \sum_{i=1}^{n-1} f_i) \; , \end{aligned}$$

which can be addressed by the gradient descent method. The components of the gradient for the transformed problem are given as:

$$\frac{\partial \tilde{C}}{\partial f_i} = \frac{\partial C}{\partial f_i} - \frac{\partial C}{\partial f_n} \quad i = 1 \ldots n-1$$

The second approach is the gradient projection method, which can be used to address both equality and inequality constraints, possibly nonlinear. Its rationale is to project solutions of the unconstrained gradient descent method into the feasible domain of a problem. More specifically, at each iteraton t, it computes the search direction $\Delta\mathcal{F}^{(t)}$

using the gradient descent method. However, the resulting solution may not be feasible, i.e. $(\mathcal{F}^{(t)} - \alpha\nabla C(\mathcal{F}^{(t)})) \notin \mathcal{FD}$. In this case, we can apply a projection function $P_{\mathcal{FD}}$ that finds the closet point in $\mathcal{FD}$ as the next step. That is,

$$\mathcal{F}_{proj}^{(t+1)} = P_{\mathcal{FD}}[\mathcal{F}^{(t)} - \alpha\nabla C(\mathcal{F}^{(t)})] \tag{3.11}$$

Let us consider an example where the feasible domain is a one-dimension interval [0,1]. All solutions $\mathcal{F}^{(t)}$ that are larger than 1 are projected to 1; All negative solutions are set as 0. In a more complex case, the projection function $P_{\mathcal{FD}}$ may have different realizations. A requirement for any realization is that the corrected step should reduce the current cost. That is $C(\mathcal{F}_{proj}^{(t+1)}) < C(\mathcal{F}^{(t)})$. However, as one may expect, the projection can be crude, from which the convergence rate may suffer.

We can also integrate the first two methods to address a problem with both equality and inequality constraints. We firstly eliminate linear equality constraints through affine transformations and then apply the gradient projection method.

In summary, the gradient projection method computes firstly the unconstrained direction, then it trims the solution to feasible. Another angle to tackle the issue is to take the constraints into account when computing search directions. In the case where both equality and inequality constraints are linear, a reduced gradient method, called the Frank-Wolfe method, has been proposed [Frank and Wolfe, 1956]. The Frank-Wolfe method applies the first-order Taylor approximation(3.9) to the original constrained optimization problem at each step:

$$\begin{aligned} &\text{minimize}_{\widetilde{\mathcal{F}}^{(t+1)}} \ \nabla C(\mathcal{F}^{(t)})^T(\widetilde{\mathcal{F}}^{(t+1)} - \mathcal{F}^{(t)}) \\ &\text{subject to: } \widetilde{\mathcal{F}}^{(t+1)} \in \mathcal{FD} \end{aligned}$$

Solving this problem gives a candidate search direction $\widetilde{\mathcal{F}}^{(t+1)}$. We can see that the only difference compared to the original steepest descent method is the added constraints. Because the constrained functions are linear, the set $\mathcal{FD}$ is a polyhedron. The objective function is an affine function with respect to $\widetilde{\mathcal{F}}^{(t+1)}$. That means, the original problem

reduces to a linear programming problem, which can be easily solved by algorithms such as the simplex method [Dantzig, 1963]. The final search step of Frank-Wolfe method is given as: $\mathcal{F}^{(t+1)} = \mathcal{F}^{(t)} + \alpha(\widetilde{\mathcal{F}}^{(t+1)} - \mathcal{F}^{(t)})$, where α is computed with line search methods. The advantage of the Frank-Wolfe method is that its search directions are always feasible. However, in practise, the convergence can be very slow.

3.3.2 Notable Results

In this subsection, we present some of the notable results from existing first-order optimal routing algorithms. We start from the optimal Wardrop equilibrium, which describes the state of network optimum. Then, we describe some of the algorithms that adaptively converge to the optimal Wardrop equilibrium. They can be viewed as a variation of the standard gradient method. Finally, we present results in distributed implementation of Wardrop equilibrium and the first-order method.

One of the early results is the Wardrop equilibrium [Wardrop, 1952], which is essentially the Nash equilibrium for traffic networks. It can be viewed as the result of selfish communications: Each communication increases the traffic over a path if the path shows a lower cost than other paths; It reduces the traffic over a path if the path shows a higher cost than other paths. Therefore, at a Wardrop equilibrium of a network, for each communication flow, costs of all utilized paths are equal. The value is no larger than those un-utilized paths. Following the selfish strategy, a Wardrop equilibrium is in general suboptimal. It has been demonstrated that, instead of using costs, a Wardrop equilibrium based on marginal costs is equivalent to the network optimum. In this thesis, we term that as the *optimal Wardrop equilibrium*. More specifically, the optimal Wardrop equilibrium can be described as:

For each communication w, given its traffic demand $\mathcal{T}_w$, all its available paths $\{\mathcal{P}_w\}$ can be numbered $1, ...N_{use}, ...N_{all}$ so that:

$$\frac{\partial C(\mathcal{F})}{\partial f_1} = \frac{\partial C(\mathcal{F})}{\partial f_2} = ... = \frac{\partial C(\mathcal{F})}{\partial f_{N_{use}}} = M_w \leq \frac{\partial C(\mathcal{F})}{\partial f_{N_{use}+1}} \leq ... \leq \frac{\partial C(\mathcal{F})}{\partial f_{N_{all}}}$$

$$f_i > 0, \quad i = 1, ..., N_{use}$$

$$f_i = 0, \quad i = N_{use} + 1, ..., N_{all}$$

Then the network solution is optimal, i.e. $\mathcal{F} = \mathcal{F}^*$. The optimal Wardrop equilibrium is the necessary and sufficient condition of network optimum. In order to reach such an equilibrium, each communication flow needs only to increase traffic on paths with low marginal cost, and to reduce traffic to no less than zero on paths with high marginal cost. Formally speaking,

$$f_i^{(t+1)} = [f_i^{(t)} - \alpha(\frac{\partial C(\mathcal{F})}{\partial f_i} - \frac{\partial C(\mathcal{F})}{\partial f_*})]^+ \quad i = 1...N_{all},$$

where $\frac{\partial C(\mathcal{F})}{\partial f_*} = \min\{\frac{\partial C(\mathcal{F})}{\partial f_i} \mid i = 1...N_{all}\}$ is the least first derivative cost of all w's available paths. It can be easily seen that this is essentially the projected gradient descent method with equality constraints eliminated by a affine transformation. α can be determined in a heuristic manner to avoid oscillations of solutions. Different algorithms may have a different choice on the heuristic factor [Fischer et al., 2006, Gupta and Kumar, 1997, Borkar and Kumar, 2003, Raghunathan and Kumar, 2009]. Another type of first-order method to address the constrained optimization, namely the Frank-Wolfe algorithm, has also been used to converge the optimal Wardrop equilibrium [Dafermos and Sparrow, 1969].

The (optimal) Wardrop equilibrium, in the field of telecommunication, describes a network state with respect to each communication flow, i.e. between each S-D pair. Algorithms based on the optimal Wardrop equilibrium are therefore flow-level distributed. In their seminal paper [Cantor and Gerla, 1974], Cantor and Mario first established a optimal equilibrium from telecommunication point of view. They have also devised algorithms that adaptively converges to the optimum. Later, Gallager [Gallager, 1977] has established a necessary condition and a sufficient condition for a network optimum with respect to each intermediate node, that is, a node-level distributed optimal Wardrop equilibrium. In the following, we will briefly describe the result.

Denote the traffic generated at node i addressed for node j as $\mathcal{T}_i(j)$. Let $\phi_{i,k}(j)$ be the ratio of outgoing traffic of node i that is sent through node k addressed for node j. Then, the necessary condition for the network optimum is

for each node i, and each destination node $j \neq i$, and link $(i,k) \in \mathcal{L}$,

$$\frac{\partial C(F)}{\partial \phi_{i,k}(j)} \begin{cases} = \gamma_{i,j}, & \text{if } \phi_{i,k}(j) > 0 \\ \geq \gamma_{i,j}, & \text{if } \phi_{i,k}(j) = 0 \end{cases} \tag{3.12}$$

where $\gamma_{i,j}$ is some constant value - an equivalence to M_w in the flow-level distributed distributed Wardrop equilibrium. The sufficient condition for the network optimum is

for each node i, and each destination node $j \neq i$, and any link $(i,k) \in \mathcal{L}$,

$$C'_{i,k}(f_{i,k}) + \frac{\partial C(\mathcal{F})}{\partial \mathcal{T}_k(j)} \begin{cases} = \zeta_{i,j}, & \text{if } \phi_{i,k}(j) > 0 \\ \geq \zeta_{i,j}, & \text{if } \phi_{i,k}(j) = 0 \end{cases} \tag{3.13}$$

where $C_{i,k}(f_{i,k})$ is the cost over the link (i,k) subjected to link traffic $f_{i,k}$ and $C'_{i,k}(f_{i,k}) = \frac{dC_{i,k}(f_{i,k})}{df_{i,k}}$ is the first derivative cost of the link. $\zeta_{i,j}$ is some contant for each link. It is worth noting that in the sufficient equilibrium (3.13), $\frac{\partial C(\mathcal{F})}{\partial \mathcal{T}_k(j)}$ - first derivative cost to the traffic generated at node k - is used as opposed to $\frac{\partial C(\mathcal{F})}{\partial f_k(j)}$- the first derivative cost to all traffic sending out from k to j[2]. $\frac{\partial C(\mathcal{F})}{\partial \mathcal{T}_k(j)}$ is essentially the merged first derivative cost of all paths from node i to node j. It can be calculated hop by hop as

$$\frac{\partial C(\mathcal{F})}{\partial \mathcal{T}_k(j)} = \sum_l \phi_{k,l}(j)\left[C'_{k,l}(f_{k,l}) + \frac{\partial C(\mathcal{F})}{\partial \mathcal{T}_l(j)}\right] \tag{3.14}$$

which is a proportional average of marginal costs through all potential paths.

Adaptive algorithms can be applied to converge to the sufficient condition. That is, each node i decreases traffic through a next hop k if $C'_{i,k}(f_{i,k}) + \frac{\partial C(\mathcal{F})}{\partial \mathcal{T}_k(j)}$ is large and correspondingly increases the traffic through a next hop k if $C'_{i,k}(f_{i,k}) + \frac{\partial C(\mathcal{F})}{\partial \mathcal{T}_k(j)}$ is small. Segall and Sidi [Segall and Sidi, 1981] extend the basic adaptive algorithm to better

[2]This is because that $f_k(j)$ is the sum of traffic generated at node k, i.e. $D_k(j)$ and traffic that node k relays for other nodes. The value $\frac{\partial C(\mathcal{F})}{\partial f_{k,j}}$ includes both $\frac{\partial C(\mathcal{F})}{\partial D_k(j)}$ and first derivative cost of upstream traffic beyond the control of node k

handle topological changes. The value of first-derivative costs $C'_{i,k}(f_{i,k})$ of each link (i,k) can be estimated by the perturbations technique [Cassandras et al., 1990, Guven et al., 2004].

Route discovery and maintenance are not considered in general by existing optimal routing algorithms. A common choice is to leave an open design space for potential route discovery algorithms. At each node i, a set of neighbours, denoted as $B_i(j)$, are blocked from forwarding i's traffic to destination j. Optimal routing algorithms aim to achieve the distributed optimal Wardrop equilibrium on the unblocked neighbours. This method is also known as the blocking technique. In existing optimal routing algorithms, the selection and maintenance of the blocking set $B_i(j)$ are preliminarily addressed. For example, it can be based on the rule of strictly decreasing derivative costs along intermediate nodes to a destination node [Gallager, 1977,Segall and Sidi, 1981], which affects the convergence of the gradient descent method. Furthermore, they can not guarantee loop-freedom in the presence of topology changes, which is left for multipath routing algorithms to address. In theory any multipath algorithms that discover all potential paths can be used in conjunction with a first-order optimal approach. However, as we have discussed previously in section 2.5, in order to approach a network optimum under practical considerations, it requires a multipath algorithm that is specially designed for optimal routing.

One notable example is the feasibility-condition based multipath routing algorithm MPDA [Vutukury and Garcia-Luna-Aceves, 1999]. The feasibility condition-based multipath routing algorithm can be used to discover and maintain multiple loop-free paths at all time. In addition, the averaged merging of derivative costs(3.14) is approximated with the minimum derivative cost of next-hops:

$$\frac{\partial C(\mathcal{F})}{\partial \mathcal{T}_k(j)} = \min_l \{C'_{k,l}(f_{k,l}) + \frac{\partial C(\mathcal{F})}{\partial \mathcal{T}_l(j)}\} \tag{3.15}$$

As we have discussed, such a choice establishes an order of intermediate nodes and eliminates loops. Another advantage of this approximation is a faster convergence than

algorithms using averaged merging. Equation (3.14) implies that the first derivative cost, and correspondingly the load distribution of upstream nodes are affected by the load distribution choice of downstream nodes. For example, node i distributes traffic over its neighbours based on $C'_{i,k}(f_{i,k}) + \frac{\partial C(\mathcal{F})}{\partial \mathcal{T}_k(j)} \quad \forall k \in \mathcal{N}_i$, where $\frac{\partial C(\mathcal{F})}{\partial \mathcal{T}_k(j)}$ is depended on k's distribution $\phi_{k,l}(j), \quad \forall l \in \mathcal{N}_k$. The approximation (3.15) decouples this connection, therefore sees a faster convergence than algorithms based on (3.14). However, because costs are approximated, the final solution is suboptimal.

Gallager's result has been proposed for wired network with link costs, i.e. $C_{i,k}(f_{i,k})$. However, given an analytical cost model of wireless communication that meets the condition of convex increasing, first-order methods can be used to optimize wireless routing. For example, Xi and Yeh have derived a cost model based on the signal-to-interference-plus-noise ration (SINR) and M/M/1 queueing model [Xi and Yeh, 2006]. A gradient descent method is used with this cost model. The major challenge of first-order methods in wireless domain lies in the design of cost models rather than routing algorithms.

3.4 Netwon-based Methods

In this subsection, we discuss routing algorithms that are based on Newton's method. Newton's method is a second-order algorithm. It provides fast convergence compared to first-order methods. Meanwhile, despite this merit, Newton's method imposes great challenges to the design of a distributed routing algorithm. We first present the mathematical background of Newton's method.

3.4.1 A Mathematical Background of Newton's Methods

Similar to the pattern we describe the first-order gradient-based method, we will first describe the rationale of Newton's method in an unconstrained case. Then we will discuss separately techniques to tackle equality constraints and inequality constraints.

3.4.1.1 Unconstrained Optimization

Newton's method is a second-order optimization approach in the sense that it uses the second-order Taylor approximation of the cost function at each iteration. Namely,

$$\begin{aligned} C(\mathcal{F}^{(t+1)}) &= C(\mathcal{F}^{(t)} + \Delta\mathcal{F}^{(t)}) \\ &\approx C(\mathcal{F}^{(t)}) + \nabla C(\mathcal{F}^{(t)})^T \Delta\mathcal{F}^{(t)} + \frac{1}{2}\Delta\mathcal{F}^{(t)^T} \nabla^2 C(\mathcal{F}^{(t)}) \Delta\mathcal{F}^{(t)} \end{aligned} \tag{3.16}$$

where $\nabla^2 C(\mathcal{F}) = \begin{pmatrix} \frac{\partial^2 C(\mathcal{F})}{(\partial f_1)^2} & \cdots & \frac{\partial^2 C(\mathcal{F})}{\partial f_1 \partial f_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 C(\mathcal{F})}{\partial f_n \partial f_1} & \cdots & \frac{\partial^2 C(\mathcal{F})}{(\partial f_n)^2} \end{pmatrix}$ is the matrix of second partial derivatives, i.e. hessian matrix, of the cost function. In order to minimize the objective cost function, we need to make the change of the cost as negative as possible, i,e. to minimize

$$C(\mathcal{F}^{(t+1)}) - C(\mathcal{F}^{(t)}) = \nabla C(\mathcal{F}^{(t)})^T \Delta\mathcal{F}^{(t)} + \frac{1}{2}\Delta\mathcal{F}^{(t)^T} \nabla^2 C(\mathcal{F}^{(t)}) \Delta\mathcal{F}^{(t)}$$

The right-hand side of the above equation is essentially a quadratic function with respect to $\Delta\mathcal{F}^{(t)}$, whose minimum value value occurs at

$$\Delta\mathcal{F}^{(t)} = -\nabla^2 C(\mathcal{F}^{(t)})^{-1} \nabla C(\mathcal{F}^{(t)})$$

This gives the Newton direction. Therefore, at each iteration of Newton's method

$$\mathcal{F}^{(t+1)} = \mathcal{F}^{(t)} + \alpha\Delta\mathcal{F}^{(t-1)} = \mathcal{F}^{(t-1)} - \alpha\nabla^2 C(\mathcal{F}^{(t)})^{-1} \nabla C(\mathcal{F}^{(t)})$$

The step size α in Newton's method can be decided in the same way as in the gradient descent method: constant, predefined decreasing and dynamic computed. In addition, during the convergence process, once $\mathcal{F}^{(t)}$ is within a certain neighbourhood of the optimal solution, more specifically if $\|\nabla C(\mathcal{F}^{(t)})\|_2 < \eta$ for certain positive η, Newton method takes $\alpha = 1$ and converges quardatically to the optimal solution. This stage is called the *quadratically convergent phase.* The stopping criterion of Newton's method is based on a value called the Newton decrement λ, which is defined at iteration t from

$$\lambda^2 = \Delta\mathcal{F}^{(t)^T} \nabla^2 C(\mathcal{F}^{(t)}) \Delta\mathcal{F}^{(t)} = -\nabla C(\mathcal{F}^{(t)})^T \Delta\mathcal{F}^{(t)}$$

It can be shown that $\lambda^2/2$ is an second-order approximation of $\mathcal{F}^{(t)} - \mathcal{F}^*$. Newton's method stops if $\lambda^2/2 < \epsilon$ for a predefined small positive ϵ

In addition to the fast convergence rate, another important feature of Newton's method is affine-invariant. That is, the minimization over $\mathcal{F} = A\mathcal{Z} + b$ has the same convergence as the minimization over $\mathcal{Z}$

$$\text{minimize}_{\mathcal{F}} C(\mathcal{F}) \iff \text{minimize}_{\mathcal{Z}} \; C(A\mathcal{Z} + b)$$

This is an important property that enables us to manipulates the handle variables without reducing convergence rate. As we have discussed previously, through a proper affine transformation, linear equality constraints can be eliminated from a problem. In next subsection, however, we present a different angle to tackle the equality constraints. The technique is equivalent to an affine transformation but does not require us to find the affine transformation matrix $\tilde{A}$ and a solution $\hat{\mathcal{F}}$ to $A\mathcal{F} = b$.

3.4.1.2 Equality-Constrained Optimization

We shall gradually add constraints towards an optimal routing problem. In this scene, we consider an optimization problem where only linear equality constraints are present, that is

$$\begin{aligned} &\text{minimize } y(x) \\ &\text{subject to } Ax = b \end{aligned} \tag{3.17}$$

where $A : \mathbb{R}^{cm \times n}$, $x \in \mathbb{R}^{n \times 1}$, $b \in \mathbb{R}^{cm \times 1}$. The necessary and sufficient optimality condition of the problem (3.17) is that there exist a column vector $\nu \in \mathbb{R}^{cm \times 1}$ such that

$$\begin{cases} \nabla y(x) + A^T \nu = 0 \\ Ax = b \end{cases} \tag{3.18}$$

Here we use the term $y(x)$ instead of $C(\mathcal{F})$ because we are not applying the optimality condition to the original problem. Once again, we can approximate the objective cost

function at each iteration t with a second-order Taylor expansion and try to minimize the change of cost, subject to the equality constraints. We have

$$\begin{aligned} &\text{minimize}_{\Delta\mathcal{F}^{(t)}} \ \nabla C(\mathcal{F}^{(t)})^T \Delta\mathcal{F}^{(t)} + \frac{1}{2}\Delta\mathcal{F}^{(t)^T} \nabla^2 C(\mathcal{F}^{(t)}) \Delta\mathcal{F}^{(t)} \\ &\text{subject to } A(\mathcal{F}^{(t)} + \Delta\mathcal{F}^{(t)}) = b \end{aligned}$$

which is the minimization of a quadratic function with respect to $\Delta\mathcal{F}^{(t)}$, in the form of (3.17). Differentiating the quadratic function in $\Delta\mathcal{F}^{(t)}$ gives the gradient, i.e. $\nabla y(x)$ in (3.18), as $\nabla^2 C(\mathcal{F}^{(t)})\Delta\mathcal{F}^{(t)} + \nabla C(\mathcal{F}^{(t)})$. With trivial modification, the system (3.18) can be transformed to

$$\begin{pmatrix} \nabla^2 C(\mathcal{F}^{(t)}) & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathcal{F}^{(t)} \\ \nu \end{pmatrix} = - \begin{pmatrix} \nabla C(\mathcal{F}^{(t)}) \\ 0 \end{pmatrix} \tag{3.19}$$

where $\nu \in \mathbb{R}^{cm \times 1}$ is a dual variable introduced for cm equality constraints. (3.19) is a linear equation system with $cm+n$ variables and $cm+n$ equations. It is usually referred to as the *KKT system*. At each iteration, the equality-constrained Newton's method solves the system (3.19), which gives the Newton direction $\Delta\mathcal{F}^{(t)}$ and an auxiliary variable ν. It follows the same stopping criterion as Newton's method in the unconstrained case.

It is important to note that one premise of the system (3.19) is that the current solution is already feasible, i.e. $A\mathcal{F}^{(t)} = b$. This assumption, however, may not always hold in practise, due to an infeasible initial solution or a changing environment. Adding the discrepancy between the current solution and the constraints to $\Delta\mathcal{F}^{(t)}$, it follows

$$\begin{pmatrix} \nabla^2 C(\mathcal{F}^{(t)}) & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathcal{F}^{(t)} \\ \nu \end{pmatrix} = - \begin{pmatrix} \nabla C(\mathcal{F}^{(t)}) \\ A\mathcal{F}^{(t)} - b \end{pmatrix} \tag{3.20}$$

If the step size $\alpha = 1$, then one step will correct the solution back to feasible domain. If the step size $\alpha < 1$, then it may take multiple iterations to draw the solution feasible. In the domain of the optimization theory, system (3.20) is called Newton's method with infeasible start.

3.4.1.3 Inequality-Constrained Optimization

If both equality and inequality constraints are presents in an optimization problem, the equality constraints can be addressed using the techniques described previously. In the following, we describes techniques to address purely the inequality constraints in Newton's method. Similar to approaches in first-order methods, the inequality constraints can be addressed via two types of manners: 1) to project an unconstrained step to feasible domain; 2) to compute a search direction while considering constraints.

The projection method that we have described in subsection 3.3.1 can be used in conjunction with Newton's method. Assuming $P_{\mathcal{FD}}[\cdot]$ is the projection function of the feasible domain, the projected Newton's method can be described as:

$$\mathcal{F}_{proj}^{(t+1)} = P_{\mathcal{FD}}[\mathcal{F}^{(t)} - \alpha \nabla^2 C(\mathcal{F}^{(t)})^{-1} \nabla C(\mathcal{F}^{(t)})]$$

In case only non-negative inequality constraints are presents in an optimization problem, a notable result can be found to reduce the complexity of the projected Newton's method [Bertsekas, 1981]. For the following problem

$$\begin{aligned} &\text{minimize } C(\mathcal{F}) \\ &\text{subject to } \mathcal{F} \succeq 0 \end{aligned} \tag{3.21}$$

where "$\succ$" is an element-wise comparison, Bertsekas proposes to identify the binding constraints during iterations. A binding constraint of one element f_i indicates that the current solution has stepped on the boundary of a feasible domain, i.e. $f_i = 0$ but still has an urge to decrease, i.e. $\partial C(\mathcal{F})/\partial f_i > 0$. If a constraint is binding, we can simplify the computation of Newton's method to consider less in that dimension. More specifically, define the set of indices of binding constraints $\mathcal{I}^+ = \{i \mid {f_i}^{(t)} = 0 \;\&\; \frac{\partial C(\mathcal{F}^{(t)})}{\partial {f_i}^{(t)}} > 0\}$. $D^{(t)}$ is a reduced inverse hessian, where off-diagonal elements are set as zero

$$D_{i,j}^{(t)} = 0 \quad \forall i \in \mathcal{I}^+,\ j = 1...n$$

The step of the projected Newton's method is written as:

$$\mathcal{F}_{proj}^{(t)} = [\mathcal{F}^{(t)} - \alpha D^{(t)} \nabla C(\mathcal{F}^{(t)})]^+$$

where $[\cdot]^+$ is the projection to positive orthant. Because part of $D^{(t)}$ is reduced to diagonal elements only instead of a full inverse of the hessian matrix, the computation complexity of the projected Newton's method is reduced. It has been demonstrated that the projected Newton's method retain the superlinear convergence rate of the classical Newton's method.

Through affine transformation, the result for non-negative constraints can be extended to optimization problems with box constraints, i.e.

$$\text{minimize } C(\mathcal{F})$$

$$\text{subject to } a_{low} \leq A\mathcal{F} \leq a_{high}$$

Although the constraints are more complex, the main motivation remains the same: identifying the binding constraints to reduce the computational complex at each iteration. This type of methods is also known as the active set algorithms. Many recent progress can be found in the domain of active set algorithms [Facchinei et al., 1998,Hager and Zhang, 2006].

In contrary to the projected Newton's method, another angle to tackle the inequality constraints is to bring the constraints into the computation of search directions in Newton's method. The barrier method introduces "punishments" on the overall costs in order to prevent the violation of inequality constraints. Consider the following problem,

$$\begin{aligned} \text{minimize} \quad & C(\mathcal{F}) \\ \text{subject to} \quad & g_i(\mathcal{F}) \leq 0 \quad i = 1...m \\ & A\mathcal{F} = b \end{aligned}$$

A common choice is to add the logarithm barrier function $\Phi(\mathcal{F}) = -\mu \sum_{i=1..m} log(-g_i(\mathcal{F}))$ to the objective function. The inequality constraints can be removed from the problem.

$$\begin{aligned} &\text{minimize } C(\mathcal{F}) - \mu \sum_{i=1}^{m} log(-g_i(\mathcal{F})) \qquad (3.22) \\ &\text{subject to } A\mathcal{F} = b \end{aligned}$$

The barrier function increases to infinity if $g_i(\mathcal{F}) > 0$. The scalar value μ is a small positive number that controls the "punishments". Newton's method is used to solve the equality constrained problem. The "punishments" keep each Newton iteration within the original feasible domain. At each iteration, the search direction is computed as

$$\begin{pmatrix} \nabla^2 C(\mathcal{F}^{(t)}) + \nabla^2 \Phi(\mathcal{F}^{(t)}) & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta \mathcal{F}^{(t)} \\ \nu \end{pmatrix} = - \begin{pmatrix} \nabla C(\mathcal{F}^{(t)}) + \nabla \Phi(\mathcal{F}^{(t)}) \\ 0 \end{pmatrix} \tag{3.23}$$

where

$$\nabla \Phi(\mathcal{F}^{(t)}) = -\mu \sum_{i=1}^{m} \frac{\nabla g_i(\mathcal{F}^{(t)})}{g_i(\mathcal{F}^{(t)})}$$

$$\nabla^2 \Phi(\mathcal{F}^{(t)}) = \mu \sum_{i=1}^{m} \frac{1}{g_i(\mathcal{F}^{(t)})^2} \nabla g_i(\mathcal{F}^{(t)}) \nabla g_i(\mathcal{F}^{(t)})^T - \mu \sum_{i=1}^{m} \frac{1}{g_i(\mathcal{F}^{(t)})} \nabla^2 g_i(\mathcal{F}^{(t)})$$

In case the inequality constraints are linear, the above hessian of the barrier function reduces to its first term: $\nabla^2 \Phi(\mathcal{F}^{(t)}) = \mu \sum_{i=1}^{m} \frac{1}{g_i(\mathcal{F}^{(t)})^2} \nabla g_i(\mathcal{F}^{(t)}) \nabla g_i(\mathcal{F}^{(t)})^T$ Because the added logarithm barrier function is non-zero and increasing within the feasible domain. The solution computed from the barrier method is an approximation of the optimal solution. As μ approaches to zero, the solution approaches to the optimum. The *interior-point method* introduces an outer iteration that solves the problem (3.22) with a diminishing u.

Algorithm 2 Algorithm of the interior-point method

Starting with an initial value $x^{(0)}$, $\mu = \mu^{(0)} > 0$, $0 < \gamma < 1$ and $\epsilon > 0$ **repeat**:

1. Find the solution $\tilde{\mathcal{F}}^*$ to the equality constrained problem (3.22) using Newton's method
2. Update the solution: $\mathcal{F} = \tilde{\mathcal{F}}^*$
3. Update the control scalar: $\mu = \gamma\mu$

unitl the stopping criterion is satisfied: $m\mu < \epsilon$

The stopping criterion $m\mu$ can be shown as the optimality gap between the solution of Newton's method and the optimal solution of the original constrained problem.

The interior-point method involves an inner loop of the equality-constrained Newton's method and an outer loop that decreases u to zero. Despite the nested iterations, the interior-point method retain the superlinear convergence rate of Newton's method and is often superior to the projected Newton's method. A more advanced form of the interior-point method is called the *primal-dual interior-point method.* We omit the deduction of the method, but simply give its Newton's step as follows:

$$\begin{pmatrix} \nabla^2 C(\mathcal{F}^{(t)}) + \sum_{i=1}^{m} \lambda_i \nabla^2 g_i(\mathcal{F}^{(t)}) & Dg(\mathcal{F}^{(t)})^T & A^T \\ -\text{diag}(\lambda) Dg(\mathcal{F}^{(t)}) & -\text{diag}(g(\mathcal{F}^{(t)})) & 0 \\ A & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathcal{F}^{(t)} \\ \Delta\lambda^{(t)} \\ \Delta\nu^{(t)} \end{pmatrix}$$
$$= \begin{pmatrix} \Delta C(\mathcal{F}^{(t)}) + Dg(\mathcal{F}^{(t)})^T \lambda^{(t)} + A^T \nu^{(t)} \\ -\text{diag}(\lambda^{(t)}) g(\mathcal{F}^{(t)}) - (1/\mu)\mathbf{1} \\ A\mathcal{F}^{(t)} - b \end{pmatrix} \tag{3.24}$$

where $Dg(\mathcal{F}) = \{\nabla g_1(\mathcal{F}), ..., \nabla g_m(\mathcal{F})\}^T$, $\mathbf{1} = \{1, .., 1\}^T$. $\lambda \in \mathbb{R}^{m \times 1}, \nu \in \mathbb{R}^{cm \times 1}$ are dual variables introduced for m inequality-constraints and cm equality-constraints. At each round, the $n + m + cm$ equations with $n + m + cm$ variables can be solved for an update $\Delta\mathcal{F}^{(k)}, \Delta\lambda^{(t)}$ and $\Delta\nu^{(t)}$. Replacing (3.22) with (3.24) We refer interested readers to other books for detailed descriptions [Boyd and Vandenberghe, 2004].

3.4.1.4 Iterative Computation of Newton's Direction

The Newton step at each iteration $H\Delta\mathcal{F} = b$ can be large and very expensive to solve analytically. A common approach to use numerical methods to iterative compute the solution of the linear equation system. Some of the widely used iterative methods include: conjugate gradient (CG), preconditioned conjugate gradient (PCG) [Shewchuk, 1994] and matrix splitting techniques such as the Jacobi method. If the matrix H is diagonally dominant and sparse, a recent developed iterative method called Gaussian belief propagation [Shental et al., 2008] has been demonstrated to converge fast. In addition to reduce the computation complexity, some of the method can be exploited for

distributed implementation. For example the GaBP and certain form of matrix splitting technique can be implemented in a distributed manner, which we will cover later. We refer interested readers to [Boyd and Vandenberghe, 2004] for more details.

3.4.1.5 Diagonal Approximation of the Hessian Matrix

In Newton's method, the inverse of the Hessian matrix $\nabla^2 C(\mathcal{F})^{-1}$ is the primary source of computation and coupling between each component. A practical simplification is to approximate the Hessian $\nabla^2 C(\mathcal{F})$ with its diagonal elements only, i.e. diag $(\nabla^2 C(\mathcal{F}))$. To see this, consider a diagonal approximation to the unconstrained Newton step:

$$\begin{cases} \frac{\partial^2 C(\mathcal{F})}{(\partial f_1)^2} \Delta f_1 = -\frac{\partial C(\mathcal{F})}{\partial f_1} \\ \vdots \\ \frac{\partial^2 C(\mathcal{F})}{(\partial f_2)^2} \Delta f_2 = -\frac{\partial C(\mathcal{F})}{\partial f_n} \end{cases}$$

In this way, the inverse operation is trivial. More importantly, components are separated from each other: each variable f_i can be computed by one equation. Compared to iterative methods, such as the conjugate gradient method, a diagonal approximation of the Hessian matrix can not guarantee the quadratic convergence rate of the classical Newton's method. However, it offers a great deal of practical simplicity. It exhibits a superior convergence in comparison to the gradient descent method.

3.4.2 Notable Results

Newton's method shows a great advantage over first-order methods in convergence rate. Extensive studies can be found in applying Newton's method to telecommunications, which dates back to early 1980s. Nonetheless, it remain an ongoing study as many open issues exist. In the following, we will present and discuss some of the most notable optimal routing algorithms.

3.4.2.1 Projected Newton's Method

The projected Newton's method [Bertsekas, 1981] can be used to address problems with linear inequality constraints. In a later work [Bertsekas and Gafni, 1983], Bertsekas and Gafni models network routing as:

$$\begin{aligned} &\text{minimize} \quad C(\mathcal{F}) \\ &\text{subject to} \quad \mathcal{F} \succ 0 \\ &\qquad \sum_{p \in \mathcal{P}_w} f_p = \mathcal{T}_w \quad \forall w \in \mathcal{W} \end{aligned} \tag{3.25}$$

where $\mathcal{W}$ is a set of communications and $\mathcal{T}_w$ is the traffic demand of a communication w. The equality constraints are eliminated by an affine transformation. The non-negative constraints are addressed by a projected Newton's method. Off-diagonal elements of the Hessian matrix, $\nabla^2 \tilde{C}(\mathcal{Z})$, at position (i,j), $i \neq j$ are set to zero is if either path i or j meet certain criterion. The criterion is as follows: th path is not in use but has a positive first derivative cost. This off-diagonal elimination reduces part of the computation load in solving Newton's step. The remaining non-zero off-diagonal elements couples the different communication in the computation of Newton's step. Therefore, it is a centralized algorithm. The conjugate gradient method is used to iteratively solve each Newton's step.

The approach exhibits a faster convergence in primary Newton step compared to gradient-based routing algorithms. The major drawback of this approach is that it is a centralized algorithm. The conjugate gradient method introduces multiple iterative steps to each Newton step, which requires large scale communications. The capacity constraints are absent, which leads to potential overflow of traffic at intermediate nodes.

3.4.2.2 Second-Order Extension to Gallager's Algorithm

Recall that Gallager's theorem [Gallager, 1977] establishes the sufficient and necessary conditions for an optimal solution at each node, as opposed to the classical optimal

Wardrop equilibrium for each communication flow. It essentially divides the network optimization into a number of optimization at each node. We can formulate each subproblem as node i minimizes network costs by adjusting its traffic distribution, $f_{i,k}(j)$ over each neighbour k to each destination j, namely

$$\begin{aligned} &\text{minimize}_{\substack{\Delta f_{i,k}(j)^{(t)}, \\ k\in\mathcal{N}_i,\ j\in\mathcal{N}}} \quad C(\mathcal{F}^{(t+1)}) - C(\mathcal{F}^{(t)}) \\ &\text{subject to} \quad f_{i,k}(j)^{(t)} + \Delta f_{i,k}(j)^{(t)} \geq 0, \quad \forall k, j \end{aligned}$$

Based on the formulation of Gallager's result [Gallager, 1977], Bertsekas et al. [Bertsekas et al., 1984] applies second-order Taylor expansion to the objective function and translate the subproblem at node i for a given destination j into

$$\begin{aligned} \text{minimize}_{\Delta\phi_i(j)^{(t)}} \ & \nabla_{i,j} C(\mathcal{F}^{(t)})^T \Delta\phi_i(j)^{(t)} + \bar{f}_i^{(t)} \phi_i(j)^{(t)^T} \nabla^2_{i,j} C(\mathcal{F}^{(t)}) \phi_i(j)^{(t)} \\ \text{subject to} \ & \phi_i(j)^{(t)} + \Delta\phi_i(j)^{(t)} \geq 0, \\ & \sum_{k\in\mathcal{N}_i} \Delta\phi_{i,k}(j) = 0, \quad \phi_{i,k} = 0, \forall k \in B_i(j) \end{aligned}$$

where $\bar{f}_i$ is the aggregated traffic at node i, $\phi_i(j) = \{\phi_{i,k}(j) | k \in \mathcal{N}_i\}^T$ is a column vector of traffic ratio over each outgoing link of node i addressed for destination j. $\nabla_{i,j} C(\mathcal{F}) = \{C'_{i,k}(f_{i,k}) + \frac{\partial C(\mathcal{F})}{\partial \mathcal{T}_k(j)} | k \in \mathcal{N}_i\}^T$ is a column vector of first derivative costs of paths from i to j. Correspondingly, $\nabla^2_{i,j} C(\mathcal{F})$ is the matrix of second derivatives. Its diagonal elements are $C''_{i,k}(f_{i,k}) + \frac{\partial^2 C(\mathcal{F})}{(\partial \mathcal{T}_k)^2}$. The authors replace $\nabla^2_{i,j} C(\mathcal{F}^{(t)})$ with its diagonal approximation $\text{diag}(\nabla^2_{i,j} C(\mathcal{F}^{(t)}))$. The Newton step for each node can be locally computed as:

$$\begin{cases} \phi_{i,k}(j)^{(t+1)} = \max\{0, \phi_{i,k}(j)^{(t)} - \frac{\alpha}{\bar{f}_i^{(t)}} \frac{C'_{i,k}(f_{i,k}^{(t)}) + \frac{\partial C(\mathcal{F}^{(t)})}{\partial \mathcal{T}_k(j)} - w_i(j)}{C''_{i,k}(f_{i,k}^{(t)}) + \frac{\partial^2 C(\mathcal{F}^{(t)})}{(\partial \mathcal{T}_k(j))^2}}\} \quad \forall k \in \mathcal{N}_i \\ \sum_{k\notin B_i(j)} \phi_{i,k}(j)^{(t+1)} = 1 \end{cases} \tag{3.26}$$

The equality constraints are eliminated by introducing a dual variable $w_i(j)$. Non-negative constraints are addressed by a projection method, i.e. $\max\{0, ...\}$. This equation system consists of $|\mathcal{N}_i| + 1$ variables, $\phi_{i,k}(j)^{(t+1)} \quad k = 1...|\mathcal{N}_i|$ and $w_i^{(t+1)}$, and

$|\mathcal{N}_i| + 1$ equations. $C'_{i,k}(\mathcal{F}), C''_{i,k}(\mathcal{F})$ are link cost information. $\frac{\partial C(\mathcal{F})}{\partial \mathcal{T}_k(j)}$ is path derivative cost, given by equation (3.14). $\frac{\partial^2 C(\mathcal{F})}{(\partial \mathcal{T}_k(j))^2}$ is the second derivative cost of path k. It has been demonstrated that the node-level distributed optimal routing algorithm remains convergent [Bertsekas et al., 1984].

It is worthy noting that the node-level distributed manner comes from the division of the network optimization problem into subproblems. In fact, this division resembles a *primal decomposition technique*, which we will present in section 3.5. The diagonal approximation of the Hessian matrix avoids the usage of $\frac{\partial^2 C(\mathcal{F})}{\partial \mathcal{T}_k(j) \partial \mathcal{T}_l(j)}$, $\forall k, l \in \mathcal{N}_i$, for each node i, which can be communicationally expensive to gather.

$$\frac{\partial^2 C(\mathcal{F})}{\partial \mathcal{T}_k(j) \partial \mathcal{T}_l(j)} = \sum_{(a,b) \in \mathcal{L}} q_{a,b}(k,j) q_{a,b}(l,j) C''_{a,b}(f_{a,b}) \tag{3.27}$$

where $q_{a,b}(k,j)$ is the portion of traffic $\mathcal{T}_k(j)$ that travels through link (a,b). It involves information exchange between node i and all links connecting i and j. The information required for second derivative costs can not be merged as equation (3.14) for first derivative cost. Therefore, a large amount of information needs to be exchanged. Avoiding the use of $\frac{\partial^2 C(\mathcal{F})}{\partial \mathcal{T}_k(j) \partial \mathcal{T}_l(j)}$ in Newton's step is the major merits of the diagonal approximation.

In fact, if the diagonal elements of the Hessian, $\frac{\partial^2 C(\mathcal{F})}{(\partial \mathcal{T}_k)^2}$, are computed using the general formula 3.27, large scale information exchanges are still required. Instead, the authors proposes to lower bounds of $\frac{\partial^2 C(\mathcal{F})}{(\partial \mathcal{T}_k(j))^2}$, which simplifies the computation but slows down the convergence.

In summary, the major issue left open is the computation of the second derivative path costs. Without a proper merge, it requires a large amount of information exchanges to compute true second derivative costs.

Another issue of the algorithm is inherited from Gallager's algorithm. Upstream nodes are affected by the load distribution of downstream nodes. More specifically, the Newton's step for each node, as in equation (3.26), is dependent on its overall flow $\bar{f}_i^{(t)}$. That means, the subproblem is not solved. Each node computes one step then the

subproblem becomes a new one. It is easily seen that the convergence pattern is altered from a classic Newton's method.

3.4.2.3 Interior-Point Method in Network Optimization

Routing algorithms that we have discussed so far in this section apply certain projections to Newton's method. Compared to projection methods, the interior-point Newton's method shows a superior convergence in addressing inequality constraints and is suitable for large scale optimization problems.

Zymnis et al. apply a specific interior point method, namely a primal-dual interior-point method, to the network utility maximization problem(3.7) [Zymnis et al., 2007]. A preconditioned conjugate gradient method (PCG) is used to solve Newton step in a truncated manner. A satisfying convergence rate is demonstrated via statistical evalutions - for large networks with 2×10^5 links, the algorithm converges within 25 iterations.

This approach, however, should be seen as a proof-of-concept result for the iterior-point method in networking: It operates in a centralized manner. In the following, we will present algorithms that address the distributed issue of interior-point method. In this subsection, we first discuss an approach called the *matrix splitting technique*. In next subsection, we describe the *Gaussian belief propagation method*.

3.4.2.4 Matrix Splitting Technique

In this subsection, we describe two algorithms that apply matrix splitting technique to achieve distributed optimization. Wei et al. develop a matrix splitting method that can be used to decouple components of a linear equation system and apply this method to a network utility maximization problem for a flow-level distributed algorithm [Wei et al., 2010]. Jia et al. apply Wei's matrix splitting method to a different formulation, which gives a node-level distributed optimal routing algorithm.

1) A matrix splitting method for distributed NUM algorithm

Wei et al. propose a distributed Newton method for network utility maximization.

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{i=S} U_i(\mathcal{T}_i) \\ \text{subject to} \quad & \mathcal{R}\mathcal{T} \leq \mathcal{C} \\ & \mathcal{T} \succ 0 \end{aligned} \tag{3.28}$$

With the help of auxiliary variables $\mathcal{E}$ such that $\mathcal{R}\mathcal{T} + \mathcal{E} = \mathcal{C}$, the problem can be transformed into non-negative constraints and equality constraints. Applying the interior-point method, the problem becomes:

$$\begin{aligned} \text{minimize} \quad & y(x) = -\sum_{i=1}^{S} U_i(x_i) - \mu \sum_{i=1}^{S+L} log(x_i) \\ \text{subject to} \quad & Ax = \mathcal{C} \end{aligned}$$

where $x = \{\mathcal{T}, \mathcal{E}\}$, $A = \{\mathcal{R}, I\}$. According to equation (3.19) and (3.23), this equality constrained problem can be addressed by iterative steps:

$$\begin{cases} \Delta x^{(t)} = -\nabla^2 y(x^{(t)})^{-1}(\nabla y(x^{(t)}) + A^T \nu^{(t)}) & (3.29) \\ (A\nabla^2 y(x^{(t)})^{-1} A^T)\nu^{(t)} = -A\nabla^2 y(x^{(t)})^{-1}\nabla y(x^{(t)}) & (3.30) \end{cases}$$

Since the utility function $U_i(\mathcal{T}_i)$, $i = 1...S$ is defined on each single-path communication, $y(x)$ is separable with respect to each x_i, $i = 1...S+L$. Both $\nabla^2 y(x)$ and $\nabla^2 y(x)^{-1}$ are diagonal. The primal step (3.29) is separable therefore can be solved locally at each source node and each link, given the value of the dual variable $\nu^{(t)}$. However, the dual step (3.30) is not separable and can not be directly addressed in a distributed manner. The major contribution of Wei's distributed Newton method is to apply a matrix splitting technique to solve the dual step distributedly. For the purpose of convenience, denote $G = A\nabla^2 y(x^{(t)})^{-1} A^T$ and $b = -A\nabla^2 y(x^{(t)})^{-1}\nabla y(x^{(t)})$. The dual step (3.30) is written as

$$G\nu^t = b$$

where G is a $L \times L$ matrix, $\nu^{(t)}$ and b are both $L \times 1$ vectors.

Matrix splitting is a type of iterative method to solve the above equation system. It splits $G = M + N$ and solves

$$M\tilde{\nu}^{(k+1)} = b - N\tilde{\nu}^{(k)} \tag{3.31}$$

If a matrix splitting approach converges, as $k \to \infty$, the value of $\tilde{\nu}^k$ approaches $\nu^{(t)}$, which is the solution of equation (3.30). In fact, the Jacobi method and the Gauss-Seidel method can be classified as matrix splitting approaches.

Wei et al. propose to split G into three parts: 1) a diagonal matrix D with diagonal element $(D)_{i,i} = (G)_{i,i}$; 2) a residual matrix $B = G - D$; 3) a diagonal matrix $\bar{B}$ with diagonal element $\bar{B}_{i,i} = \sum_{j=1}^{L}(B)_{i,j}$ Therefore, we have

$$G = \underbrace{(D + \bar{B})}_{M} + \underbrace{(B - \bar{B})}_{N}$$

where$D + \bar{B}$ is diagonal.

Wei's algorithm is based on a preliminary attempt [Jadbabaie et al., 2009]. It has been proved that as long as the matrix splitting approach converges to a certain neighbourhood of the solution to dual step, i.e. $\nu^{(t)} \pm \epsilon$ for some small error ϵ, then the interior-point Newton's method converges quadratically to a neighbourhood of the network optimum. In a subsequent work [Wei et al., 2011], the convergence of the matrix splitting technique is proven and analysed.

To illustrate how this approach works in detail, let us consider an example as shown in Fig. 3.4. As shown in the model (3.28), the objective is to maximize $U_1(\mathcal{T}_1) + U_2(\mathcal{T}_2)$. The corresponding matrices $\mathcal{R}, A, x$ are:

$$\mathcal{R} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \quad A = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix} \quad x = \begin{pmatrix} \mathcal{T}_1 \\ \mathcal{T}_2 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix}$$

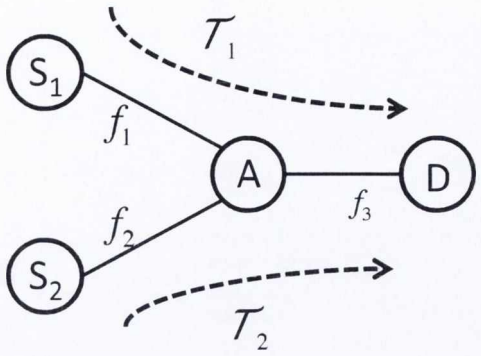


Fig. 3.4: Consider two single path communications $\mathcal{T}$ and $\mathcal{T}_2$,which are originated from node S_1 and S_2 respectively. Node D is their destination. Denote traffic over three links as f_1, f_2 and f_3 respectively

where $r_1 = \mathcal{C}_1 - \mathcal{T}_1$, $r_2 = \mathcal{C}_2 - \mathcal{T}_2$ and $r_3 = \mathcal{C}_3 - \mathcal{T}_1 - \mathcal{T}_2$ are the auxiliary variables for capacity residual at each link. The transformed objective function $y(x)$ is

$$y(x) = -\sum_{i=1}^{2} U_i(x_i) - \mu \sum_{i=1}^{5} log(x_i)$$

The components of this objective function include 2 sources and 3 links. Its gradient and inverse of Hessian are as follows,

$$\nabla y(x) = \begin{pmatrix} -U_1'(x_1) - \frac{\mu}{x_1} \\ -U_2'(x_2) - \frac{\mu}{x_2} \\ -\frac{\mu}{x_3} \\ -\frac{\mu}{x_4} \\ -\frac{\mu}{x_5} \end{pmatrix} \quad \nabla^2 y(x)^{-1} = \begin{pmatrix} (-U_1''(x_1) + \frac{\mu}{x_1^2})^{-1} & 0 & 0 & 0 & 0 \\ 0 & (-U_2''(x_2) + \frac{\mu}{x_2^2})^{-1} & 0 & 0 & 0 \\ 0 & 0 & \frac{\mu}{x_3^2} & 0 & 0 \\ 0 & 0 & 0 & \frac{\mu}{x_4^2} & 0 \\ 0 & 0 & 0 & 0 & \frac{\mu}{x_5^2} \end{pmatrix}$$

For the purpose of convenience, we denote individual element in $\nabla y(x)$ as g_i and denote diagonal element in $\nabla^2 y(x)^{-1}$ as h_i, where i is the index of elements. It should be noted that g_i and h_i can be computed at each component with only local information.

Accordingly, we have

$$
\begin{aligned}
G &= A\nabla^2 y(x)^{-1} A^T \\
&= \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} h_1 & 0 & 0 & 0 & 0 \\ 0 & h_2 & 0 & 0 & 0 \\ 0 & 0 & h_3 & 0 & 0 \\ 0 & 0 & 0 & h_4 & 0 \\ 0 & 0 & 0 & 0 & h_5 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} h_1 + h_2 & 0 & h_1 \\ 0 & h_2 + h_4 & h_2 \\ h_1 & h_2 & h_1 + h_2 \end{pmatrix}
\end{aligned}
$$

The splitting of G gives:

$$
M = \begin{pmatrix} 2h_1 + h_3 & 0 & 0 \\ 0 & 2h_2 + h_4 & 0 \\ 0 & 0 & 2h_1 + 2h_2 + h_5 \end{pmatrix} \quad N = \begin{pmatrix} -h_1 & 0 & h_1 \\ 0 & -h_2 & h_2 \\ h1_1 & h_2 & -h_1 - h_2 \end{pmatrix}
$$

Based on these matrices, according to equation (3.31), the dual step $\nu^{(k+1)}$ can be computed as:

$$
\begin{cases} \tilde{\nu}_1^{(k+1)} = \frac{h_1\tilde{\nu}_1^{(k)} - h_1\tilde{\nu}_3^{(k)} - h_1 g_1 - h_3 g_3}{2h_1 + h_3} \\ \tilde{\nu}_2^{(k+1)} = \frac{h_2\tilde{\nu}_2^{(k)} - h_2\tilde{\nu}_3^{(k)} - h_2 g_2 - h_4 g_4}{2h_2 + h_4} \\ \tilde{\nu}_3^{(k+1)} = \frac{(h_1+h_2)\tilde{\nu}_3^{(k)} - h_1\tilde{\nu}_1^{(k)} - h_2\tilde{\nu}_2^{(k)} - h_1 g_1 - h_2 g_2 - h_5 g_5}{2h_1 + 2h_2 + h_5} \end{cases} \tag{3.32}
$$

The primal step $\Delta x^{(t+1)}$ can be computed given the final solution to the above iterations $\nu^{(t)}$,

$$
\begin{cases} \Delta\mathcal{T}_1 = \Delta x_1 = -(h_1 g_1 + h_1\nu_1^{(t)} + h_1\nu_3^{(t)}) \\ \Delta\mathcal{T}_2 = \Delta x_2 = -(h_2 g_2 + h_2\nu_2^{(t)} + h_2\nu_3^{(t)}) \end{cases} \tag{3.33}
$$

In summary, Wei's algorithm includes three layers of iterations. The outer iteration is the interior-point iteration, as shown in Algorithm 2. The middle iteration is the

equality-constrained Newton iteration. Each source node starts with an initial feasible traffic $\mathcal{T}_1^{(0)}$ and $\mathcal{T}_2^{(0)}$. With traffic transmitted, at each round t, each source node can calculate g_i, h_i. These values are forwarded to all links of its corresponding path. The inner iteration is the matrix splitting method. Given g_i, h_i from source nodes, each link iteratively updates the dual step $\tilde{\nu}_i^{(k)}$. At each step of the inner iteration, each link share its current $\tilde{\nu}_i^{(k)}$ with all links that running the same flow. That is, information exchange between link 1 and 3, between link 2 and 3. A new step is computed as stated in (3.32). Once $\tilde{\nu}_{(i)}^{(k)}$ converges to $\nu_i^{(t)}$, current inner iteration stops. Links send this information back to source nodes. The primal step at each source is computed as stated in (3.33).

As shown in the example, Wei's algorithm is flow-level distributed. It uses interior-point to address inequality constraints. The major contribution is the application of a matrix-splitting technique to solve the Newton step. It has been proven to converge superlinearly at the Newton iteration. However, although the matrix splitting technique proposed can decouple an arbitrary matrix, Wei's algorithm relies on the separable structure of the objective function. That is, it addresses a single path utility maximization problem. The matrix splitting method introduces extra iterations for each Newton step. More importantly, at each step of inner iteration, all links belong to the same flow need to share their dual step $\tilde{\nu}^{(k)}$. It incurs frequent large scale information exchanges.

2) Applying Wei's matrix splitting to a node-level distributed optimal routing algorithm

Let us now investigate further on the major issue of Wei's algorithm. In Wei's algorithm, the coupling dual step (3.30) is introduced from the equality-constraints. In the previous example, there are three equality-constraints: At each of the three links, the traffic runs through the link plus the residual of the capacity equals the capacity of the link. A dual variable is introduced for each constraint, ν_1, ν_2 and ν_3 respectively. That means, the three duals are coupled via the mutual flows through them. The matrix splitting technique essentially replaces the coupling in computation, say between $\tilde{\nu}_1^{(k+1)}$

and $\tilde{\nu}_3^{(k+1)}$, with the copy of previous step, i.e. $\tilde{\nu}_1^{(k)}$ and $\tilde{\nu}_3^{(k)}$, as in the first equation of (3.32). In this way, computation of the dual step can be executed locally at each link, but all coupled links need to exchange their $\tilde{\nu}_i$ at each iteration.

Following Wei's result, Liu and Sherali develop a multipath routing algorithm that maximizes utility function of each communication flow [Liu and Sherali, 2012]. Instead of source flow $\mathcal{T}_i$, they use link rate for each flow $f_{(i,j)}^w$ as handle variables. Equality-constraints are introduced from flow reservation at each node: the sum of outgoing traffic - to other nodes or to application layers - equals the sum of incoming traffic - from other nodes or from application layers. Each dual variable ν_i is only coupled with neighbours who it shares links with. As opposed to Wei's algorithms, information exchanges happen between neighbour nodes in the matrix splitting iterations. The major contribution is applying the approach of Wei's algorithm to the different formulated problem, which gives node-level distributed routing algorithm.

However, this approach achieves its distributed manner by introducing more variables and thus more equality-constraints. For instance, for the simple network shown in Fig. 3.4, a formulation using link rate for each flow, as suggested by Liu and Sherali can be given as:

$$\begin{aligned} \text{maximize } & U_1(\mathcal{T}_1) + U_2(\mathcal{T}_2) \\ \text{subject to } & f_1^1 = \mathcal{T}_1; \;\; f_2^2 = \mathcal{T}_2 \\ & f_3^1 = f_1^1; \;\; f_3^2 = f_2^2 \\ & 0 \leq f_1^1 < \mathcal{C}_1 \\ & 0 \leq f_2^2 < \mathcal{C}_2 \\ & 0 \leq f_3^1 + f_3^2 < \mathcal{C}_3 \end{aligned}$$

Applying the same affine transformation as Wei's algorithm and using interior-point

method to eliminate non-negative constraints, the problem can be translated into:

$$\begin{aligned}
\text{minimize } & -U_1(\mathcal{T}_1) - U_2(\mathcal{T}_2) - \mu\Big(log(f_1^1) + log(f_2^2) \\
& + log(f_3^1) + log(f_3^2) + log(r_1) + log(r_2) + log(r_3)\Big) \\
\text{subject to } & f_1^1 = \mathcal{T}_1;\ \ f_2^2 = \mathcal{T}_2 \\
& f_3^1 = f_1^1;\ \ f_3^2 = f_2^2; \\
& f_1^1 + r_1 = \mathcal{C}_1 \\
& f_2^2 + r_2 = \mathcal{C}_2 \\
& f_3^1 + f_3^2 + r_3 = \mathcal{C}_3
\end{aligned}$$

It is a problem with 9 handle variables and 7 equality-constraints. Adding one dual variable for each constraint, the linear equation system of Newton's step is 7×7, compared to flow-level distributed step with 3×3 system. It can be easily seen that, translating the handle variable from flow rate to link rate for each flow, the size of the linear system at Newton's step increases from the number of links $|L| \times |L|$ to

$$(|L| + \sum_{w \in \mathcal{W}} \sum_{p \in P_w} |p|) \times (|L| + \sum_{w \in \mathcal{W}} \sum_{p \in P_w} |p|)$$

where $|p|$ is the length of a path p. Although it can be argued that the increased computation overhead imposes little challenge to moderate devices, a larger equation system incurs more iterations to the matrix splitting method at each Newton step. In particular, for a wireless network where our interests lies, flows and links are coupled to a great extend. It is unclear whether this distributed algorithm can gain any benefits compared to Wei's flow-level distributed algorithm. Unfortunately, their results lack of statistical or analytical justification. Furthermore, their routing algorithm relies on the separable structure of the objective function and can not be applied to cost minimization. Also, a utility maximization routing may likely lead to excessively long paths and large delays [Zymnis et al., 2007].

$$\begin{cases} G_{1,1}x_1 + G_{1,2}x_2 + G_{1,3}x_3 + G_{1,4}x_4 = b_1 \\ G_{2,1}x_1 + G_{2,2}x_2 + G_{2,3}x_3 + 0x_4 = b_2 \\ G_{3,1}x_1 + G_{3,2}x_2 + G_{3,3}x_3 + G_{3,4}x_4 = b_3 \\ G_{4,1}x_1 + 0x_2 + G_{4,3}x_3 + G_{4,4}x_4 = b_4 \end{cases} \Rightarrow$$

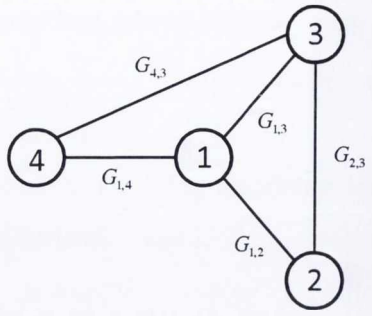


Fig. 3.5: An example of constrain graph: the linear equation system to the left can be translated to a constrain graph to the right. In the GaBP algorithm, Each node starts with a parameter set $\{b_i/G_{i,i}, G_{i,i}\}$ and neighbours iteratively exchanges messages $\Lambda_{i,j}$.

3.4.2.5 Gaussian Belief Propagations

In the following, we present the rationale and application of an iterative method, namely the Gaussian belief propagation (GaBP) method, in a distributed network optimization algorithm. Belief propagation (BP) is a message passing algorithm to conduct inference on probabilistic graphical models. GaBP is a type of BP algorithm where the underlying probability density function (PDF) the Gaussian function, that is, a normal distribution.

The GaBP algorithm can be used to distributedly solve a linear system $Gx = b$, on the premise that G is symmetric [Shental et al., 2008]. The rationale of applying GaBP algorithm is as follows: We can construct a joint normal distribution whose mean value is the optimal solution of linear equation system, i.e. $x = G^{-1}b$. Such a construction can be described by a function $p(x)$:

$$p(x) = \mathcal{Z}^{-1}e^{-x^TGx/2+b^Tx} = \mathcal{N}(\mu, G^{-1})$$

where $\mu = G^{-1}b$ and $\mathcal{Z}$ is some factor matrix. In order to implement in a distributed manner, we need to find separately the mean value of the marginal distribution for each component, namely $p(x_i)$. This can be achieved with a BP algorithm, through inference between components.

The operation of GaBP can be described with a graphical model, as shown in Fig. 3.5.

Each component x_i corresponds to a node in the graph, which represents a priori normal distribution. The distribution is specified by an inherent parameter set $\{\mu_{i,i},\ \Lambda_{i,i}^{-1}\}$, namely $\mathcal{N}(\mu_{i,i} = b_i/G_{i,i},\ \Lambda_{i,i}^{-1} = \delta_i^2 = G_{i,i})$, where $\mu_{i,i} \in \mathbb{R}$ is the mean and $\Lambda_{i,i} \in \mathbb{R}$ is the inverse of the variance. The coupling relation between two components x_i and x_j, which exists if $G_{i,j} = G_{j,i} \neq 0$, corresponds to an undirected link connecting node i and node j. The link represents an exponential function $e^{-x_i G_{i,j} x_j}$, with an inherent parameter $G_{i,j}$.

At each round k, each node constructs and shares messages with all its neighbours. The message sent from node i to node j is a parameter set $\{\mu_{i,j}^{(k)}, \Lambda_{i,j}^{(k)}\}$ of a joint normal distribution. Starting from $\Lambda_{m,i}^{(0)} = 0, \mu_{i,j}^{(0)} = 0$, the parameter set is updated from inherent parameters of nodes and links, as well as previously received messages:

$$\Lambda_{i,j}^{(k+1)} = -G_{i,j}^2 (\Lambda_{i,i} + \sum_{\substack{m \in N_i \\ m \neq j}} \Lambda_{m,i}^{(k)})^{-1}$$

$$\mu_{i,j}^{(k+1)} = -G_{i,j}^{-1} (\Lambda_{i,i}\mu_{i,i} + \sum_{\substack{m \in N_i \\ m \neq j}} \Lambda_{m,i}^{(k)} \mu_{m,i}^{(k)})$$

Once the algorithm converges, the mean value of the marginal distribution for component x_i is given as:

$$\mu_i = (\Lambda_{i,i} + \sum_{\substack{m \in N_i \\ m \neq j}} \Lambda_{m,i}^{(*)})^{-1} (\Lambda_{i,i}\mu_{i,i} + \sum_{\substack{m \in N_i \\ m \neq j}} \Lambda_{m,i}^{(*)} \mu_{m,i}^{(*)})$$

which is the value of x_i in the solution $x = G^{-1}b$.

It should be noted that, although the theory behind the GaBP is the probability inference, the algorithm itself is a message-exchange algorithm that operates purely with the parameters of nodes, links and messages. We are interested in its application to distributed network optimization problem. For more details of the GaBP algorithm, such as convergence condition, we refer to a recent study [Bickson, 2009].

Inspired by its fast convergence and distributed nature, Bickson et al. [Bickson et al., 2009] apply the GaBP algorithm to the centralized primal-dual interior-point method-based network optimization algorithm [Zymnis et al., 2007]. At each Newton's step, the

GaBP algorithm is used to solve (3.24) iteratively and distributedly. Each component corresponds to a single path communication flow w. The inherent parameters $b_{i,i}/G_{i,i}$ and $G_{i,i}^{-1}$ of each component can be retrieved locally.

Like the centralized NUM algorithm [Zymnis et al., 2007], the objective function in the distributed NUM algorithm [Bickson et al., 2009] is separable utility function for each communication flow. Because the Hessian matrix is only part of the matrix G and that the GaBP algorithm decouples each components with regards G, the same distributed algorithm can be applied to coupled objective function without modifications. $G_{i,j}$ can be get where a flow shared capacity with another flow.

The distributed NUM algorithm is flow-level distributed algorithm, as the GaBP decouples components - different communication flow in this case. The authors suggest broadcast of messages from one flow to all its coupled flows. One of its problems is that two flows may be coupled with each other at different links. That means, one link in the probabilistic graphical model, as shown in Fig. 3.5 may consist of multiple physical locations. Broadcasting at all locations may lead to duplication and confusions. Either by unicast or as suggested by broadcast, it is yet to be addressed how to share messages in an actual networking algorithm.

3.4.3 Summary of Newton's Method

Newton's method is an iterative method to optimize convex problems. At each step, it approximates the objective function with a second-order Taylor expansion. Compared to first-order methods, Newton's method exhibits a superior convergence rate. The convergence of Newton's method is affine-invariant. Linear equality constraints can be eliminated by affine-transformation. Or equivalently, through the KKT system(3.19). Inequality constraints can be addressed by projection method. Especially if the inequality constraints are linear, projection method is a simple yet satisfying approach. A more advanced choice is the interior-point method. The basic interior-point method introduces a barrier function for each inequality constraint. In order to keep solution within

the feasible domain of a problem, barrier functions introduce extra penalty as a solution approaches infeasible domain. Interior-point method shows a faster and "smoother" convergence than projection method in dealing with inequality constraints.

We have reviewed some of the notable routing algorithms that uses Newton's method. NUM algorithms based on Newton's method are also included in our review. The NUM problem resembles the cost minimization routing problem in their mathematical models, with primarily difference in the objective function. Many results of NUM algorithms can be applied to the optimal routing problem with little modifications.

Both the projected Newton's method and the interior-point method can be applied to routing optimization problem. However, by its nature form, Newton's method is a centralized approach. The major challenge to the design of a distributed Newton's method lies in the computation of the Newton step, which is essentially a linear equation system $Gx = b$, at each iteration. Some iterative methods have been proposed to address this challenge. We have reviewed two of them that shows great promises. Wei et al. develop a matrix splitting technique that is convergent for an arbitrary square matrix G. They apply this technique to a flow-level distributed NUM algorithm. Another iterative method that facilitates the distributed computation of a linear equation system is the Gaussian belief propagation. Compared to many other iterative methods, the GaBP algorithm has been shown with a promising convergence rate. Bickson et al. has developed a flow-level distributed NUM algorithm based on the GaBP method. Both algorithms lacks of practical considerations in the sense that they either incurs large amount of communication overhead or ignored the message exchange mechanism completely. However, it is important to extract the mathematical merits from the actual algorithms: Both Wei's matrix splitting technique and Bickon's GaBP method decouple the computation of each component x_i using coupled value from the previous round.

We have also seen some node-level distributed routing algorithms. Bertsekas et al. apply Newton's method to Gallager's model, which was initially proposed for distributed optimal Wardrop equilibrium. Jiu and Sherali apply Wei's matrix splitting technique

to a multipath utility maximization problem with link rate of each communication flow $f^w_{(i,j)}$as handle variable. Both approaches face issues of slowed convergence compared to their flow-level or centralized counterparts but lack of statistical or analytical evaluations. It remains an open issue whether existing node-level optimization algorithms provide a satisfying convergence. If not, how can we design a node-level optimal routing algorithm that retains the convergence rate of a flow-level or centralized optimization algorithms.

3.5 Decomposition Methods

In previous section, we have described algorithms based on Newton's method, which enjoy a superlinear convergence rate. The major issue of Newton-based algorithms is that the computation is centralized by nature. Although there are some promising iterative approaches that facilitate a distributed Newton's method, these progresses are quite recent and we have a small amount of network algorithms that apply these techniques. On the other hand, a standard approach for distributed optimization algorithms is the *decomposition method.*

There are primarily two types of decomposition methods: the primal decomposition and the dual decomposition. Both types of methods divide an optimization problem into a number of subproblems each of which can be solved separately and locally. The coupling between different subproblems are represented by "interfaces" - primal variables in primal decomposition and dual variables in in dual decomposition. During the computation of each subproblem, these interfaces treated as a constant. Once all subproblems complete their computation, based on their results, a new set of values are given to the interfacing primal or dual variables. Decomposition methods can be viewed as a mix of a second-order optimization for each subproblem and a first-order optimization for computing interfacing variables.

Compared to iterative methods in distributed Newton computation, decomposition methods are in general simple to implement. In particular, the dual decomposition

method have an economical interpretation: dual variables between subproblems can be interpreted as prices for resources. Because both the optimal routing problem and the utility maximization problem can be regarded as a type of resource allocation problems, decomposition methods, especially the dual decomposition method, have attracted extensive research efforts in the domain of network optimizations.

In this section, we will first give an overview of the mathematical background of decomposition methods. Then we review some of network optimization algorithms that are based on decomposition techniques. Finally, we examine the relations between decomposition methods and Newton's method in order to understand the nature of distributed optimizations.

3.5.1 A Mathematical Background

In the following, we firstly give a brief description on the Lagrange dual function of a constrained optimization problem. With the concept of duality established, we discuss the decomposition techniques that have been used to decentralize optimal routing algorithms.

3.5.1.1 Lagrange Dual and the KKT Condition

The Lagrangian $L : \mathbb{R}^{n+m+cm} \to \mathbb{R}$ of the constrained optimization problem (3.4) is defined as

$$L(\mathcal{F}, \lambda, \nu) = C(\mathcal{F}) + \sum_{i=1}^{m} \lambda_i g_i(\mathcal{F}) + \sum_{i=1}^{cm} \nu_i h_i(\mathcal{F})$$

where, λ, ν are called the dual variables or Lagrange multiplier of the problem. A Lagrange dual function of the problem (3.4) $\mathcal{D} : \mathbb{R}^{m+cm} \to \mathbb{R}$:

$$\mathcal{D}(\lambda, \nu) = \inf_{\mathcal{F} \in \mathbf{dom}\ C} L(\mathcal{F}, \lambda, \nu) = \inf_{\mathcal{F} \in \mathbf{dom}\ C} \left(C(\mathcal{F}) + \sum_{i=1}^{m} \lambda_i g_i(\mathcal{F}) + \sum_{i=1}^{cm} \nu_i h_i(\mathcal{F}) \right) \quad (3.34)$$

Because $g_i(\mathcal{F}) \leq 0$, $h_i(\mathcal{F}) = 0$ for any $\mathcal{F} \in \mathcal{FD}$, for any $\lambda \succeq 0$ and any ν, we have

$$\mathcal{D}(\lambda, \nu) \leq C(\mathcal{F}^*)$$

This means that the function $\mathcal{D}(\lambda, \nu)$, given $\lambda \succeq 0$, is an lower bound of the optimal cost. We are interested in whether the above inequality is tight, that is whether the two sides *can* be equal. Denote λ^*, ν^* as the dual variables that maximize the low bound, namely

$$\mathcal{D}(\lambda^*, \nu^*) = \max \mathcal{D}(\lambda, \nu) \tag{3.35}$$

It has been established that $\mathcal{D}(\lambda^*, \nu^*) = C(\mathcal{F}^*)$ if the convex optimization problem(3.4) is strictly feasible, i.e. there exists a feasible $\mathcal{F} \in \mathcal{FD}$ so that $g_i(\mathcal{F}) < 0$ for $i = 1...m$. The value $C(\mathcal{F}^*) - \mathcal{D}(\lambda^*, \nu)$ is refered to as the *duality gap*. That mean, as long as a convex problem is strictly feasible, we can solve it by addressing the following problem:

$$\begin{aligned} &\text{maximize}_{\lambda,\,\nu}\ \mathcal{D}(\lambda, \nu) \\ &\text{subject to}\ \ \lambda \succeq 0 \end{aligned} \tag{3.36}$$

In the context of optimization theory, the problem (3.36) is known as the dual problem, correspondingly the problem (3.4) is called the primal problem.

3.5.1.2 Decomposition of Optimization Problems

Decomposition methods divide an optimization problem into a number of subproblems. Each subproblem can be addressed separately. This enables a distributed implementation of optimization algorithms. A master problem sits on top of and optimizes the coordination of all subproblems, as shown in Fig. 3.6. Firstly, with an intial setting, all subproblems are computed. Their information, sometimes termed as the *price*, is reported to the master problem. The master problem is computed with prices of all subproblems, returning a new setting to each subproblem.

There are mainly two types of decomposition methods: A primal decomposition method is based on the primal problem while a dual decomposition method is based on the Lagrange dual problem. As we have previously mentioned, an optimization problem can be coupled through both the objective function and the constraints, i.e. the coupling variables and the coupling constraints. Because the dual variables, i.e. λ, ν are

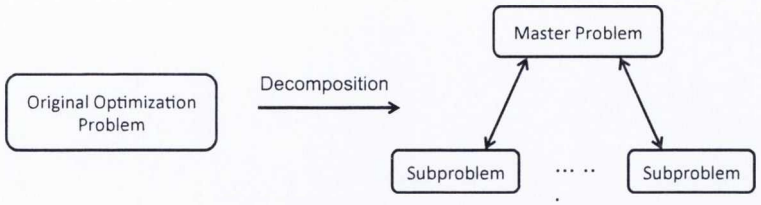


Fig. 3.6: A sketch of decomposition: The original problem is decomposed into a number of subproblems. The coordination of all subproblems forms a master problem.

introduced from the equality and inequality constraints (3.34), the dual decomposition is naturally suitable to decouple constraints while the primal decomposition is naturally suitable to decouple the objective function. However, through transformation of the problem, both decomposition methods can be used to address the two types of coupling.

Dual decomposition methods are comparatively more common in distributed optimal routing algorithms. In the following, we will firstly describe the dual decomposition technique and then the primal decomposition.

3.5.1.3 Dual Decomposition

In the following, we will describe the dual decomposition method. Firstly, we show how a dual decomposition method addresses a problem with coupling constraints. Secondly, we demonstrate that with the help of auxiliary variables, a dual decomposition method can also solve problems with coupling objective functions. Finally, we give an overview of the dual decomposition method.

Consider an optimization problem with a separable objective function and coupling

constraints:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} C_i(f_i) \\ \text{subject to} \quad & \sum_{i=1}^{n} \tilde{g}_{i,j}(f_i) \preceq \mathcal{C}_j \quad j = 1...m \\ & \sum_{i=1}^{n} \tilde{h}_{i,j}(f_i) = \mathcal{T}_j \quad j = 1...cm \\ & f_i \in \mathcal{FD}_i \quad i = 1...n \end{aligned} \tag{3.37}$$

where $\tilde{g}_{i,j} : \mathbb{R} \to \mathbb{R}$, $\mathcal{C}_j$ is the j^{th} inequality constraints. $\tilde{h}_{i,j} : \mathbb{R} \to \mathbb{R}$, $\mathcal{T}_j$ is the j^{th} equality constraints[3] The set $\mathcal{FD}_i$ is the separate feasible domain as a result of constraints on f_i only. According to the definition(3.34), the Lagrange dual function of the problem(3.37) is

$$\begin{aligned} \mathcal{D}(\lambda, \nu) &= \inf_{\forall i:\ f_i \in \mathcal{FD}^i} \Big(\sum_{i=1}^{n} C_i(f_i) + \sum_{j=1}^{m} \lambda_j (\sum_{i=1}^{n} \tilde{g}_{i,j}(f_i) - \mathcal{C}_j) + \sum_{j=1}^{cm} \nu_j (\sum_{i=1}^{n} \tilde{h}_{i,j}(f_i) - \mathcal{T}_j) \Big) \\ &= \sum_{i=1}^{n} \Big(\inf_{f_i \in \mathcal{FD}^i} (C_i(f_i) + \sum_{j=1}^{m} \lambda_j \tilde{g}_{i,j}(f_i) + \sum_{j=1}^{cm} \nu_j \tilde{h}_{i,j}(f_i)) \Big) - \sum_{j=1}^{m} \lambda_j \mathcal{C}_j - \sum_{j=1}^{cm} \nu_j \mathcal{T}_j \end{aligned}$$

The infimum of a function can be determined with a minimization of the function. Therefore the Lagrange dual function can be divided into n minimization subproblems. The maximization of the dual function is the master problem. More specifically, for each component i, the subproblem is:

$$\begin{aligned} \text{minimize} \quad & C_i(f_i) + \sum_{j=1}^{m} \lambda_j \tilde{g}_{i,j}(f_i) + \sum_{j=1}^{cm} \nu_j \tilde{h}_{i,j}(f_i) \\ \text{subject to} \quad & f_i \in \mathcal{FD}^i \end{aligned}$$

Given the set of dual variables $\lambda = \{\lambda_j\}, \nu = \{\nu_j\}$, each subproblem can be addressed by Newton's method locally and separately, returning a set of solutions $\mathcal{F}^* = \{f_i^*\}$.

[3]We wish to emphasize that this form of constraints is a special case of the general constraints in (3.5) and (3.6): $\sum_{i=1}^{n} \tilde{g}_{i,j}(f_j) - \mathcal{C}_j = g_j(\mathcal{F})$ and $\sum_{i=1}^{n} \tilde{h}_{i,j}(f_i) - \mathcal{T}_j = h_j(\mathcal{F})$. That is, the constraint functions are separable. It can be seen that the optimal routing problem fits this case. The general constraints can be addressed by adding auxiliary variables.

However, the separately computed solution set $\mathcal{F}^*$ may not be feasible, in the sense that either

$$\sum_{i=1}^{n} \tilde{g}_{i,j}(f_i^*) \succeq \mathcal{C}_j \text{ for any } j = 1...m$$

or

$$\sum_{i=1}^{n} \tilde{h}_{i,j}(f_i^*) \neq \mathcal{T}_j \text{ for any } j = 1...cm$$

Once solutions of all subproblems are computed and communicated to a central component, the infeasibility shown above may be detected. In the case of an infeasible solution, a naive projection method can be applied. Namely, we can cut off the infeasible part of $\tilde{g}_{i,j}$ and $\tilde{h}_{i,j}$ equally among all components, which gives the subproblem for each i:

$$\begin{aligned}
\text{minimize} \quad & C_i(f_i) \\
\text{subject to} \quad & \tilde{g}_{i,j}(f_i) \leq \tilde{g}_{i,j}(f_i^*) - \frac{\sum_{i=1}^{n} \tilde{g}_{i,j}(f_i^*) - \mathcal{C}_j}{n} \quad j = 1...m \\
& \tilde{h}_{i,j}(f_i) = \tilde{h}_{i,j}(f_i^*) - \frac{\sum_{i=1}^{n} \tilde{g}_{i,j}(f_i^*) - \mathcal{T}_j}{n} \quad j = 1...cm \\
& f_i \in \mathcal{FD}_i
\end{aligned}$$

Solving this subproblem of each component gives an feasible solution set $\mathcal{F}^*$. Gathering the information of the feasible solution $\mathcal{F}^*$, the master problem can be written as:

$$\begin{aligned}
& \text{maximize}_{\lambda,\nu} \quad \mathcal{D}(\lambda, \nu) = \sum_{j=1}^{m} \lambda_j a_j + \sum_{j=1}^{cm} \nu_j b_j + c \\
& \text{subject to} \quad \lambda \succeq 0
\end{aligned}$$

where $a_j = \sum_{i=1}^{n} \tilde{g}_i(f_i^*) - \mathcal{C}_j$, $b_j = \sum_{i=1}^{n} \tilde{h}_i(f_i^*) - \mathcal{T}_j$ and $c = \sum_{i=1}^{n} C_i(f_i^*)$. The master problem is convex however may not be differentiable. It is often addressed with a first-order subgradient method. Each step of the master problem gives an update of dual variables. With the new value of dual variables, each subproblem is computed again. The optimization stops when the master problem terminates.

We can also address the coupling objectives using dual decomposition methods. Consider a problem with a coupled objective function:

$$\begin{aligned} &\text{minimize } C_1(f_1, f_c) + C_2(f_2, f_c) \\ &\text{subject to } f_1 \in \mathcal{FD}_1 \\ &\qquad\qquad f_2 \in \mathcal{FD}_2 \end{aligned} \tag{3.38}$$

where f_c is the coupling variable. We can add auxiliary variables to the coupling objective function, so that coupling is transformed from the objective function to the constraints. The problem can be transformed into:

$$\begin{aligned} &\text{minimize } C_1(f_1, f_3) + C_2(f_2, f_4) \\ &\text{subject to } f_3 = f_4 \\ &\qquad\qquad f_1 \in \mathcal{FD}_1 \\ &\qquad\qquad f_2 \in \mathcal{FD}_2 \end{aligned}$$

This problem meets the form of (3.37) and can be addressed with the dual method as we have described previously.

In summary, in a decomposition method, a set of initial dual variables are given to each component. Each subproblem is computed locally, for example using Newton's method. The solutions of each subproblem are sent back to the master problem. One step of the master problem is computed using the subgradient method to updates of the dual variables. The updates are sent to subproblems for a new round of computation. An interesting interpretation of the dual variables, which is often used in the domain of resource allocation, is that they are the price of resources. Because such a price is only a soft "punishment" in the sense that it will not increase to infinity if the solution is infeasible, the solution during the iteration of dual decomposition methods may not always be feasible. Often, a certain type of projection methods are used in conjunction with dual decomposition methods, which slows down the convergence.

3.5.1.4 Primal Decomposition

In the following, we briefly describe the primal decomposition technique. As the name suggested, the primal decomposition is based on the primal problem. The idea is that each component keeps coupling variables fixed while optimizing its part of the objective function, which is a subproblem. A master problem optimizes over the coupling variables. The primal decomposition suits naturally if a problem when fixing certain variables divides into multiple subproblems. Consider again the optimization problem (3.38). Instead of adding an auxiliary variable as in the dual decomposition method, we can fix the coupling variable f_c. The original problem can be decomposed into the following subproblems, for $i = 1, 2$:

$$\begin{aligned} &\text{minimize}_{f_i} && C_i(f_i, f_c) \\ &\text{subject to} && f_i \in \mathcal{FD}_i, \end{aligned}$$

which can be addressed by Newton's method locally at each component. Given the optimal solution to each subproblem f_i^* and f_i^*, the master problem is given as:

$$\text{minimize}_{f_c} \quad C_1(f_1^*, f_c) + C_2(f_2^*, f_c)$$

which can be addressed by either the gradient method - if the differentiable - or the subgradient method - if not differentiable.

The primal decomposition technique can be used to address coupling constraints with the assist of auxiliary variables. Consider a simple example as follows:

$$\begin{aligned} &\text{minimize} && C_1(f_1) + C_2(f_2) \\ &\text{subject to} && f_1 + f_2 \leq c \\ & && f_1 \in \mathcal{FD}_1, \quad f_2 \in \mathcal{FD}_2 \end{aligned}$$

We can add an auxiliary variable to separate the the coupling constraints, so that the

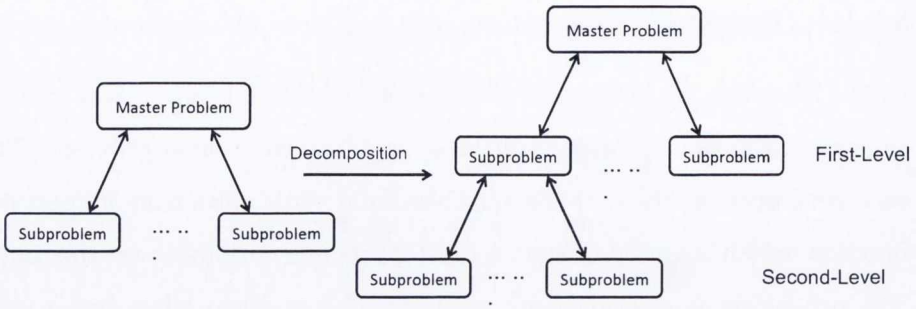


Fig. 3.7: A sketch of two-level hierarchical decomposition: Subproblems are decomposed using either the primal decomposition or the dual decomposition

problem is broken into two subproblems, for $i = 1, 2$:

$$\begin{aligned} \text{minimize} \quad & C_i(f_i) \\ \text{subject to} \quad & f_i \leq c_i, \quad f_i \in \mathcal{FD}_i \end{aligned}$$

The master problem is to find the best assignment of c_1, c_2, given $c_1 + c_2 = c$. It can be shown that the subgradient of the master problem is essentially the summation of the dual variables of each subproblems.

3.5.1.5 Hierarchical Decomposition

A hierarchical decomposition involves iterative decompositions to subproblems, as shown in Fig. 3.7. It divides a problem into multiple levels of subproblems. The condition for a convergent hierarchical decomposition is that the lower level subproblems need to run at a faster pace than higher level subproblems or master problems. This is an important design factor. For example, we can divides a network optimization into subproblems of each communication flows and divides the subproblem of each flow into subproblems of link scheduling at each intermediate hop.

In a hierarchical decomposition, both the primal decomposition and the dual decomposition can be used depending on the engineering requirements. The introduction of

auxiliary variables increase the flexibility of decomposition techniques in network optimizations. These are the mathematical tools that can be used.

3.5.2 Notable Results

In the previous subsection, we have discussed basic forms of decomposition methods. In this subsection, we are going to reviewed the application of decomposition techniques in the domain of network optimization. Extensive studies can be found in applying decomposition methods

Based on the application of decomposition techniques, we can organize the literature into two categories: vertical decompositions and horizontal decompositions. A vertical decomposition is a decomposition applied to the communication stack. It addresses joint optimization across multiple layers, including the MAC layer, the network layer or the transport layer. On the other hand, a horizontal decomposition addresses decoupling of computation between physically distant components, such as flows and nodes. That is, a horizontal decomposition facilitates distributed manner of optimization and a vertical decomposition enables cross-layer optimization.

Finally, it should be noted that our organization of the literature is not strictly a classification of existing decomposition-based algorithms. Rather, such an organization is based on the focus of each algorithm. It is possible that an algorithm includes both vertical and horizontal decompositions. In fact, the existence of layered architectures is to make communication functionalities local to physical locations that they cover, that is, to achieve distributed operations. Therefore, a vertical decomposition algorithm is likely to include certain horizontal decompositions.

3.5.2.1 Basic Dual Decomposition for Distributed NUM

Decomposition methods have been widely studied in the domain of network optimization algorithms [Chiang et al., 2007] [Palomar and Chiang, 2006]. In the following, we will describe a basic form of dual decomposition-based algorithm as well as some early

investigations on the basic algorithm.

Because of its convenience to decouple constraints, dual decomposition is a common approach to address NUM problems where objective functions are separable. The basic form is rather straightforward. Consider the following problem,

$$\begin{aligned} &\text{maximize} \quad \sum_{i=1}^{S} U_i(\mathcal{T}_i) \\ &\text{subject to} \quad \mathcal{R}\mathcal{T} < \mathcal{C}, \end{aligned}$$

where R is the single path routing matrix, namely $\mathcal{R}_{j,i} = \begin{cases} 1 & \text{if flow } i \text{ runs through link } j \\ 0 & \text{otherwise} \end{cases}$

$\mathcal{T}_i$ is the traffic of flow i. $\mathcal{C}_j$ is the link capacity of link j. Applying Lagrangian dual function, the subproblem for communication i can be written as:

$$\text{maximize}_{\mathcal{T}_i} \quad U_i(\mathcal{T}_i) - \sum_{j=1}^{m} \lambda_j \mathcal{R}_{j,i} \mathcal{T}_i$$

This subproblem can be solved by locally at the source node of each communication flow using, for example, Newton's method. Given the optimal solution $\mathcal{T}^*$, the master dual problem is given as:

$$\begin{aligned} &\text{minimize}_{\lambda} \quad \sum_{i=1}^{m} U_i(\mathcal{T}_i^*) + \lambda^T(\mathcal{R}\mathcal{T} - \mathcal{C}) \\ &\text{subject to} \quad \lambda \succ 0; \end{aligned}$$

At each link j, where the inequality constraint comes from, the dual variable can be updated by a projected subgradient method,

$$\lambda_j^{(t+1)} := [\lambda_j^{(t)} - \alpha(\mathcal{C}_j - \sum_{i=1}^{S} \mathcal{R}_{j,i} \mathcal{T}_i^*)]^+$$

Note that for link j, the value $\mathcal{C}_j - \sum_{i=1}^{n} \mathcal{R}_{j,i} \mathcal{T}_i^*$ is essentially the sum of capacity residual of the link. The dual variables λ can be interpreted as the unit price to each communication for using link resources, as it plays a negative effect on utilities of subproblems. The

NUM algorithm achieves flow-distributed using the basic dual decomposition method, which is indeed quite elegant: Start with an initial dual variable, each source node computes locally its optimal data rate. Once a link receives traffic from all communications, it updates its unit price using the subgradient method and sends the updated price to all sources whose traffic runs through the link.

In their seminal paper [Kelly et al., 1998], Kelly et al. formulates a NUM problem to achieve proportional fairness among different flows. More specifically, their consider the NETWORK problem formulated as follows:

$$\text{maximize} \sum_{i=1}^{S} w_i log(\mathcal{T}_i)$$

$$\text{subject to } \mathcal{R}\mathcal{T} \leq \mathcal{C}$$

They propose to use a subgradient method to address both the master problem and the subproblems. Their main contribution is the demonstration of stability and convergence of the distributed flow-level algorithm through a Lyapunov stability analysis.

Low and Lapsley have also investigated the basic dual decomposition technique in NUM problems [Low and Lapsley, 1999]. In addition to the synchronous algorithm stated above, they have proposed an asynchronous algorithm that allows different communications update their prices at different time intervals, which may be caused by communication delay and heterogeneous computing abilities of different nodes. Their asynchronous algorithm uses a weighted moving average as an estimate for prices at sources and for flow rates at links. They demonstrated that such an algorithm is convergent.

3.5.2.2 Utility Maximization and Multipath Routing

The majority of NUM algorithms use single path for each communication flow. The NUM problem with a multipath routing is non-trivial yet mathematically tractable. Its

general formulation is as follows.

$$\begin{aligned} &\underset{f_p:\, p\in\mathcal{P}}{\text{maximize}} \sum_{i=1}^{S} U_i(\mathcal{T}_i) \\ &\text{subject to } \mathcal{T}_i = \sum_{p\in\mathcal{P}_i} f_p, \quad m_i \le \mathcal{T}_i \le M_i, \quad i \in \mathcal{W} \\ &\mathcal{R}\mathcal{F} \le \mathcal{C}, \mathcal{F} \succ 0. \end{aligned}$$

where $\mathcal{R}$ is the routing matrix and $\mathcal{F} = \{f_p | p \in \mathcal{P}\}$ is the set of traffic of all paths, m_i and M_i are boundaries on aggregated traffic of each communication. Compared to the basic NUM problem, each communication consists of multiple paths. The separable utility function takes aggregated traffic of each communication. Its handle variables are traffic rate of each path. This problem is difficult to solve because it is no longer concave with respect to path rates even if the utility function is strictly concave with respect to aggregated traffic. Existing convex optimization methods can not be applied directly.

In order to address this non-convex problem, the objective function can be modified with the help of auxiliary variables $\tilde{f}_p$ as follows:

$$\underset{f_p,\tilde{f}_p:\, p\in\mathcal{P}}{\text{maximize}} \sum_{i=1}^{S} U_i(\mathcal{T}_i) + \sum_{i=1}^{S} \sum_{p\in\mathcal{P}_i} \frac{a_p}{2}(f_p - \tilde{f}_p)^2$$

where a_p is some constant assigned to each path. The constraints remain the same as the original problem. It is easily verified that the modified problem has the same optimal solution as the original problem. In addition, the modified problem is concave with respect to f_p and $\tilde{f}_p$ separately, although not jointly. A number of *proximal optimization approaches* have been used to address such a problem [Wang et al., 2003], [Lin and Shroff, 2006], where the dual decomposition and the subgradient method are used to solve f_p and dual variables iteratively. Along the convergence, the auxiliary variables $\tilde{f}_p$ are updated iteratively and asymptotically approaching to f_p

$$\tilde{f}_p^{\,(t+1)} = \text{updater}(\tilde{f}_p^{\,(t)}, f_p^{(t)})$$

where updater$(\cdot)$ is some arithmetic function asymptotically approaching f_p^*. For example, one possible updater is updater$(\tilde{f_p}^{(t)}, f_p^{(t)}) = \beta \tilde{f_p}^{(t)} + (1-\beta) f_p^{(t)}$ for $\beta \in [0,1)$

The convergence of such an approach have been analysed [Lin and Shroff, 2006].

3.5.2.3 Coupled Objective Functions

Although the majority of NUM algorithms consider a separable objective function, we have seen several NUM algorithms that use the dual decomposition method to solve coupled objective functions.

Tan et al. [Tan et al., 2006]consider a NUM problem where different flows are coupled through the utility function, as well as through constraints as in the basic form.

$$\underset{\mathcal{T}_i}{\text{maximize}} \sum_{i=1}^{S} U_i(\mathcal{T}_i, \{\mathcal{T}_j\}_{j \in \mathcal{N}(i)})$$

$$\text{subject to } \mathcal{T}_i \in \mathcal{FD}_i, \ \forall i; \ \sum_{i=1}^{S} g_i(\mathcal{F}) \leq \mathcal{C}$$

where the second parameter of the utility function is a set of interfering flows of each communication. Their approach to solve this problem is to introduce auxiliary variables to translate the coupling from the objective function to constraints.

$$\underset{\mathcal{T}_i, \mathcal{T}_{i,j}}{\text{maximize}} \sum_{i=1}^{S} U_i(\mathcal{T}_i, \{\mathcal{T}_{i,j}\}_{j \in \mathcal{N}(i)})$$

$$\text{subject to } \mathcal{T}_i \in \mathcal{FD}_i, \ \forall i; \ \sum_{i=1}^{S} g_i(\mathcal{F}) \leq \mathcal{C}$$

$$\mathcal{T}_{i,j} = \mathcal{T}_j, \ \forall i, j \in \mathcal{N}(i)$$

where $\mathcal{T}_{i,j}$ is a local copy of coupling traffic $\mathcal{T}_j$ at flow i. The objective function is now separable. The resulting problem can be addressed by the dual decomposition method.

He et al. [He et al., 2007] have proposed a modified objective function in order to approximate the joint optimization of utility function at transport layer and cost function at network layer.

$$\underset{\mathcal{T}_i}{\text{maximize}} \sum_{i=1}^{S} U_i(\mathcal{T}_i) - \sum_{l} C_l(\sum_{i=1}^{S} \mathcal{R}_{l,i}\mathcal{T}_i)$$

$$\text{subject to } \mathcal{R}\mathcal{T} \leq \mathcal{C}$$

where $C_l(\cdot)$ is the cost over link i. Although the utility function is separable, because the presence of link cost function, the objective function of this problem is not separable. Similar to the previous approach, He et al. introduce auxiliary variables to coupling variables and then apply dual decomposition.

Although both of the reviewed algorithms are single path communication, their present an alternative approach to existing iterative methods in addressing coupled objective function.

3.5.2.4 Node-Level Distributed Optimal Routing Algorithms

Decomposition methods can be used for node-level distributed algorithms. The authors of a recent paper [Mosk-Aoyama et al., 2010] consider network optimization problem with node rate as handle variables.

$$\text{minimize } \sum_{i=1}^{n} C_i(\bar{f}_i)$$

$$\text{subject to } A\mathcal{F} = b; \quad \mathcal{F} \succ 0$$

where $\bar{f}_i$ is data rate of node i, $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^{m\times 1}$, $A\mathcal{F} = b$ indicates the flow conservation condition for each link. Although not immediately applicable, this equality constraint takes minor modifications to facilitate a multi-commodity routing algorithm. The capacity constraint is not considered in this formulation. The objective cost function is separable with respect to its handle variables. Therefore, this problem is coupled through only equality constraints. To address this problem, firstly, Mosk-Aoyama et al. apply the interior-point method to address the non-negative constraints. Then, they use the dual decomposition method to the equality constraints, which decouples the problem.

They have also discussed the case where coupled linear inequality constraints are present, that is the equality constraints are replaced by $A\mathcal{F} \leq b$. It enables the modelling of link capacity, which is essential for optimal routing algorithms. Because the inequality constraints are coupled now, barrier functions introduced on inequality constraints are coupled. To address this, they introduce auxiliary variables $\mathcal{Z}$, such that the inequality $A\mathcal{F} \leq b$ is transformed to equality $A\mathcal{F} + \mathcal{Z} = b$ and non-negativity $\mathcal{Z} \geq 0$. The transformed problem is written as:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} C_i(\bar{f}_i) - \mu \sum_{i}^{n} log(\bar{f}_i) - \mu \sum_{i}^{m} log(z_i) \\ \text{subject to} \quad & A\mathcal{F} + \mathcal{Z} = b, \end{aligned}$$

whose objective function is separable with respect to $\bar{f}_i$ and z_i. However, the dual decomposition method can not be applied directly to such a problem: The subproblem for any given z_i is to minimize $-\mu log(z_i) + \lambda_i z_i$, whose minimum value approaches and reaches to negative infinity as λ_i approaches and reaches 0. In order to address this issue, they add an extra term $\phi(z_i)$ to the term, where $\phi(\cdot)$ is any positive function that increases faster than logarithm function, for example, a quadratic function. This remedy term avoids unbounded subproblems, however introduces a bias to their solutions.

In addition, they propose to use a distributed gossip algorithm to share information, such as primal and dual values, among different components.

In another node-level distributed algorithm [Purkayastha and Baras, 2008], the authors consider a special cost function over each link (i, j),

$$C_{i,j}(\bar{f}_{(i,j)}) = \int_0^{\bar{f}_{(i,j)}} x[D_{i,j}(x)]dx$$

where $D_{i,j}(\cdot)$ indicates the queueing delay over link (i, j) of a unit traffic. However, the motivation and the application of such an integral cost is unjustified: It gives the overall delay of all traffic in the case that they arrive at the same time. It can be easily verified that the long term overall delay for a stable queue is $\bar{f}_{(i,j)} D_{i,j}(\bar{f}_{(i,j)})$. The main

contribution of this algorithm is that the authors established an optimal equilibrium of link traffic based on the dual prices.

In essence, both approaches achieve node-level distributed manner by using link flow rates as handle variables. Indeed, this approach resembles the previously reviewed node-level distributed Newton's method [Liu and Sherali, 2012]. Similarly, using link flow rate as handle variables results in a large amount of variables and equality-constraints. In dual decomposition, each constraint introduces a dual variable, which is updated using the slowly converging subgradient method. Therefore, node-level distributed algorithms suffer from a slower convergence, compared to their flow-level counterparts.

3.5.3 Discussion of Decomposition Methods

In this subsection, we give an overview of decomposition methods as an approach for optimal routing algorithms.

The advantages of decomposition methods include 1) It decouples components of an optimization problem. If path flows are used as handle variables, flow-level distributed manner is achieved. If link flows are used as handle variables, node-level distributed manner of is achieved. 2) It has very good practical implications, i.e. prices of resources

Some of the main disadvantages of decomposition methods particularly for routing problems are as followings. 1) It may not be always feasible. Therefore, projection methods have to be used which affects the convergence. 2) Because subgradient method is commonly used to update dual or primal variables, the resulting convergence may not be strictly descent. 3) The dual decomposition is equivalent to the Newton's method with infeasible start(3.20). Instead, it uses subgradient method to address the coordination between different subproblems. Therefore, decomposition method can be seen as a mixture of first-order methods and second-order methods. It has been demonstrated that [Bickson et al., 2009, Wei et al., 2010] the GaBP and matrix splitting distributed iterative Newton's methods also enjoy a better convergence compared to decomposition methods.

3.6 Routing Optimization in Wireless Domain

Studies of optimization algorithms in wireless ad hoc networks begin to appear in recent years. Several models of optimal routing [Toumpis, 2006] [Altman et al., 2008] and [Altman et al., 2008] have been proposed. Their algorithms are based on optimal Wardrop equilibrium to achieve optimal routing. They focus on massively dense ad hoc networks, where nodes locate so close to each other that they become points in a continuum. Under such an environment, they study how data traffic running over a physical region. The main limitation of these approaches is that they do not consider neighbouring interference and do not suggest individual nodal behaviour. Instead, they utilize existing wireless throughput models and provide a macro view of traffic engineering. Therefore, it is difficult to design a routing algorithm based on these proposals.

It is easily seen that the challenge for the design of wireless optimal routing algorithms is to model the underlying wireless interference. A conflict graph [Jain et al., 2003] and [Chen et al., 2006] is a two dimension map that indicates mutual interferences of all links. With the help of conflict graph, wireless interference can be transformed into graph problems. For example, it translates the wireless scheduling problem to maximal independent sets problem or maximum weighted matching problems. Nonetheless, conflict graph itself does not model delay but merely describe the interference relations.

In his seminar paper, Bianchi [Bianchi, 2000] makes use of Markov chain to analyse the queueing and medium access behaviour of DCF. Following this idea, a more recent approach [Tickoo and Sikdar, 2008] tries to release the Poisson arrival assumption by assuming a GI/GI/1 queueing model. However, this approach assumes saturated traffic at each node, in that there are always data waiting in the queue after one successful transmission. Under the Poisson arrival assumption, Malone et al. [Malone et al., 2007] have developed an advanced analysis of the medium access delay in a heterogeneous non-saturated 802.11 DCF network.

Gupta and Shroff [Gupta and Shroff, 2011] realise the difficulty in wireless modelling.

Instead, they propose a lower bound on delay modelling using the concept of a (K, X)-bottleneck. A (K, X)-bottleneck is a subset X of a network where at most K links of it can transmit at the same time. The mean delay of a network is lower bounded by the sum of a set of mutual exclusive bottlenecks.

3.7 Miscellaneous Results

We list in the following a number of remotely related results, which can be seen as supporting techniques for optimal routing approaches.

3.7.1 Estimates of Traffic Demands

[Zhang et al., 2005] address the issues of multiple traffic matrices, i.e. changing traffic demands, by comprising solutions to facilitate multiple traffic demands as much as possible. [Dai et al., 2008] discuss the estimation traffic over a number of traffic matrices. [Xu et al., 2011] translate the network optimization problem into an entropy maximization problem, so that nodes need to distribute traffic at each hop based on an exponential entropy value.

3.7.2 Asynchronous Algorithms

[Nedic and Ozdaglar, 2009] model an optimization problem as a control system. Their results focus on demonstrating the convergence and stability of asynchronous update of cost information and decision making.

3.8 Summary and Discussion

In this section, we firstly summarize the review of this chapter. Then, we discuss open questions in the literature, which identifies the design space of an optimal routing algorithms in wireless ad hoc networks.

3.8.1 Summary of Optimal Approaches

We have reviewed in this chapter existing optimal routing algorithms and their mathematical backgrounds. Following the analysis of general multipath routing algorithm in chapter 2, existing optimal routing algorithms focus on the data distribution task of a multipath routing algorithm. They apply optimization theory to compute a traffic distribution pattern that minimizes the overall cost. The optimal routing problem can be viewed as a type of constrained convex optimization problem and optimal routing algorithms are essentially applications of optimization methods. There are several mathematical methods across different optimal routing algorithms to address different issues. In the following, we summarize optimal approaches as a whole.

3.8.1.1 Orders of Optimization Methods

Depending on the order of Taylor approximation to objective function at each iteration step, there are two types of mathematical methods to solve an optimization problem: the first-order gradient descent methods and the second-order Newton's method. In comparison, gradient descent methods are simple and distributed with respect to each handle variable by nature. However, it suffers from a slow convergence rate. Early attempts at routing optimization approaches are based on the steepest descent methods but recent progresses have been focused on its second-order counterparts. On the other hand, the Newton's method enjoys a superlinear scale-invariant convergence but requires global knowledge, i.e. the Hessian matrix. Recent results in Newton-based routing algorithms have been focus on distributed computation of Newton step.

3.8.1.2 Approaches to Address Constraints

There are a number of techniques that can be used in optimization methods to address constraints. A simplistic approach is the projection method, which can be used in conjunction with both gradient descent method and Newton's method. The projec-

tion method may result in a crude change of search direction, thus affects largely the convergence. For a linear constrained problem, Frank-Wolfe method translates the computation of gradient descent step into a linear programming problem. It finds reduced gradient descent steps that are feasible.

Because Newton's method is scale-invariant, linear equality constraints can be eliminated with affine transformation without affecting the convergence of Newton's method. A more advanced approach to address inequality constraints, compared to the projection method, is the interior-point method. It introduces a logarithm barrier function to Newton's method. The barrier function increases to infinity as a solution approaching to infeasible domain thus keeps the solution within the feasible domain. The problem for logarithm barrier function is that it addresses only strict inequalities. For example, the capacity constraint for a link flow traffic f is a strict inequality, $f < C$, which can be addressed by $log(C - f)$; However, the non-negative condition for traffic f is not strict, $f \geq 0$. By using $log(f)$, it eliminates the possibility of $f = 0$, which is not desired.

Dual Decomposition methods can be viewed as an approach to solve equality and inequality constraints. At each step of the outer iteration, they introduce a price for each constraint - more specifically the dual variable - to all subproblems. It resembles the barrier function in that the price pushes solutions of each subproblem within the feasible domain. However, in contrary to the superlinear penalty in logarithm barrier function, the penalty increases linearly at a slope of the dual variable as the solution approaching the infeasible domain. It leads to a slower convergence in addressing constraints. In addition, a linearly increasing penalty does not reach infinity at infeasible point. Therefore, dual decomposition methods do not guarantee strictly feasible step during convergence. A projection method is usually used in conjunction to enforce feasibility.

3.8.1.3 Distributed Computation of Optimization Algorithms

An optimization problem can be coupled in two ways: through the objective function and through the constraints. A distributed optimal routing algorithm requires a decou-

pled computation of each handle variable, i.e. component, in an optimization problem. Distributed computation is in general not challenging for first-order methods, because of its simplicity. A common approach is based on the optimal Wardrop equilibrium which identify the state for each communication flow at the network optimum. Many heuristic algorithms have been proposed to converge to the optimal Wardrop equilibrium. Distributed computation of second-order methods are in general challenging. There are several mathematical methods that decouple an optimization problem. They can be classified into two categories.

A standard approach is to use decomposition techniques. Decomposition methods have two basic forms: the primal decomposition and the dual decomposition. The general idea is to divide an optimization problem into a number of subproblems. Representing the coupling between subproblems, primal variables or dual variables are used to coordinate different subproblems. The task of the coordination is called the master problem, which is generally solved by a subgradient method. In network optimization algorithms, decomposition methods can used recursively to create multiple level of subproblems. The advantage of decomposition methods is that they can easily be configured to achieve different types of distributed computation. They have in general good engineering implications. For example, in certain algorithms, sum of queue length along a path can be used as a subgradient that update the master problem. The problem for decomposition methods is their slow convergence. As we have seen, part of the optimization - the master problem - is solved by a subgradient method, which is sublinear. In particular, as the number of couplings between different components increases, the convergence deteriorates quickly.

Another way to decouple the computation of an optimization problem is to use certain types of iterative methods to solve Newton step in the form of $Ax = b$. In the domain of network optimization, we have seen iterative methods such Wei's matrix splitting technique and the GaBP method. Their rationales are different: Wei's approach is an implementation of the standard matrix splitting technique; GaBP is based on a message

exchange algorithm, namely belief propagation, to compute the marginal mean of joint Gaussian distribution. Nonetheless, their basic manners resemble each other: Each component, say each communication flow, iteratively compute its estimate of the final solution based on its local information and previously computed solutions of coupling components. In comparison, Wei's matrix splitting guarantees convergence while the GaBP method requires certain convergence conditions, which is the same as the Jacobi method. If convergent, the GaBP method exhibits a faster convergence rate compared to existing matrix splitting approaches. It has been demonstrated statistically that these distributed iterative methods converge faster than decomposition methods.

These are methods that can be used by optimal routing algorithms to decouple an optimization problem. We have classified existing optimal algorithms into two levels of distributed manners: flow-level distributed algorithms and node-level distributed algorithms. The majority of optimal routing algorithms and NUM algorithms are flow-level distributed. That is, handle variables of their optimization problem represent traffic rate of a path used by a communication. Once the optimization problem is decoupled using the aforementioned mathematical tools, each communication flow can compute its traffic distribution over all its paths. There are several algorithms that are node-level distributed. They use the link rate of each flow at each node as the handle variable. Therefore, a decoupled optimization problem can be computed at each node distributively. In this way, however, the amount of variables and couplings equality-constraints increases drastically compared to flow-level distributed optimization. It can create large communication overhead and slows down the convergence of an optimal routing algorithms.

3.8.2 Open Questions

1) What is a suitable distributed approach to address the coupling introduced by equality and inequality constraints?

The first criterion for a suitable approach is that it can quickly coordinate between

different components within coupling constraints. The logarithm barrier function in the interior-point method introduces a superlinear penalty. However, it does not address the coupling between different components in constraints. Dual decomposition methods decouple constraints with respect to each component. However, their penalties are linear. Combining both methods, an ideal approach is to separately introduce a fast increasing penalty for an individual component.

The second criterion for a suitable approach is that it should guarantee feasibility at all time. The nature of routing algorithms dictates that constraints must be met. For example, the traffic of each link can never exceed its capacity. Projected methods satisfy a strictly feasible condition however suffer from distortion of convergence pattern. The logarithm barrier function guarantees only strict inequalities. It may require engineering modifications to address non-strict inequalities. Dual decomposition methods do not guarantee feasibility. We need to search for an approach that meets both criteria.

2) How can we address the coupling introduced by the objective function, namely the distributed inverse of the Hessian coefficient matrix, without using iterative methods at each Newton step?

As we have reviewed, there are some iterative approaches can be applied to distributively compute coupled Newton step, including the matrix splitting technique and the GaBP method. But they introduce a number of iterations between each Newton step, which can be expensive in ad hoc networks. The primal decomposition is another natural choice to divide coupled objective functions. By introducing auxiliary variables, one can also resort to the dual decomposition technique. However, in multipath routing algorithms the number of couplings in objective function between different paths - either of the same communication flow or of different communication flows - can be very large. As we know, in such case, decomposition approaches tend to converge very slow. Our question is, *is there an alternative approach that suits the requirement of wireless ad hoc networks?*

3) Is it possible to design a node-level distributed algorithm without increasing the size of the optimization problem?

To the best of our knowledge, existing node-level distributed optimal routing algorithms and NUM algorithms use the same mathematical approaches as their flow-level counterparts. They achieve a node-level distributed manner by using link rates as handle variables. This may lead to a much larger amount of handle variables and constraints. Although a centralized Newton's method is not affected by this transformation due to its quadratic scale-free convergence, iterative methods to calculate Newton step or the subgradient method in decomposition methods may be largely affected. A natural question arises: Given that a routing optimization problem is decoupled into subproblems with respect to all communication flows, can we design a node-level approach that retains the convergence of flow-level convergence?

Let us now explore this question a bit further. Recall the process of distributed multipath routing algorithms we have described in chapter 2: Cost information propagates from destination nodes to source nodes. Costs of multiple paths merge at intermediate nodes they meet. A tree structure is indeed established: The root is the source node and leaves are copies of the destination node. Each intermediate vertex corresponds to an intersecting node of multiple paths. Once aggregated information reaches a source node, data distribution is computed recursively from top level down to leaves. Each tree is a flow-level distributed subproblem, while each intermediate vertex computes parts of the problem rather than spouses a new decomposed subproblem. Their difference is as follows:

- In this tree-based scenario, when the one-pass recursive computation from sources to destinations completes, one exact step of the flow-level subproblem is reached.
- In a scenario where link rates are handle variables, components with respect to each handle variable affect each other. Therefore, a one-pass recursive computation from sources to destinations can not reduce costs as much as one Newton step of

flow-level distributed optimal algorithm:

1. In an iterative Newton method, such as using GaBP and the matrix splitting technique, multiple rounds of message exchanges are introduced to compute the Newton step.
2. In decomposition methods, each node-level subproblems are coordinated by a subgradient method. Therefore, the Newton step of a flow-level problem is reduced to a subgradient step.

We expect the tree-based scenario to achieve a superior performance compared to existing node-level distributed approaches. However, this scenario comes from our understanding towards a distributed multipath routing algorithm. The question is how can we enable it mathematically. The open question can be stated as:

Is it possible to design an optimal routing algorithm such that each intermediate node merges the costs of multiple paths and computes parts of the flow-level subproblems?

In addition to these three open questions regarding to optimization methods, because the majority of existing optimal routing approaches address the data distribution part of the multipath routing problem, it remains a question to integrate an optimization method with a multipath routing algorithm. Furthermore, network optimization approaches we have reviewed in this chapter assume a link-based cost function, which do not suit the wireless communication. In next chapter, we will discuss results in applying optimization approaches for wireless routing.

Part II

Proposed Solutions

Chapter 4

Wireless Medium Cost Function

In this chapter, we will search for a cost function that models the delay cost incurred by a data flow in wireless ad hoc networks, which can be used for an delay optimal routing algorithm. In the beginning, we will justify the necessity of a wireless cost function.

4.1 Motivation

The goal of our optimal routing algorithm is to minimize the delay experienced by all communication flows, i.e. to minimize $C(\mathcal{F})$. Such an objective function is well defined in wired networks using link cost, namely, $C(\mathcal{F}) = \sum_{i\in\mathcal{L}} C_i(\bar{f}_i) = \sum_{i\in\mathcal{L}} \ell_i(\bar{f}_i)\bar{f}_i$, where $\bar{f}_i$ is the sum of all traffic running through the link i, i.e. $\bar{f}_i = \sum_{\forall p\in\mathcal{P}:i\in p} f_p$. In an optimal routing algorithm, routing decisions of a communication flow are made based on the first and the second derivative costs of available paths, which are defined as

$$\frac{\partial C(\mathcal{F})}{\partial f_p} = \sum_{i\in p} \frac{\partial C_i(\bar{f}_i)}{\partial f_p} = \sum_{i\in p} \Big(\ell'(\bar{f}_i)\bar{f}_i + \ell_i(\bar{f}_i)\Big) \tag{4.1a}$$

$$\frac{\partial^2 C(\mathcal{F})}{(\partial f_p)^2} = \sum_{i\in p} \frac{\partial^2 C_i(\bar{f}_i)}{(\partial f_p)^2} = \sum_{i\in p} \Big(\ell_i''(\bar{f}_i)\bar{f}_i + 2\ell_i'(\bar{f}_i)\Big) \tag{4.1b}$$

$$\frac{\partial^2 C(\mathcal{F})}{\partial f_p \partial f_q} = \sum_{\substack{i\in p\\ i\in q}} \frac{\partial^2 C_i(\bar{f}_i)}{\partial f_p \partial f_q} = \sum_{\substack{i\in p\\ i\in q}} \Big(\ell_i''(\bar{f}_i)\bar{f}_i + 2\ell_i'(\bar{f}_i)\Big) \tag{4.1c}$$

where $\ell_i'(\bar{f}_i)$ and $\ell_i''(\bar{f}_i)$ is the first and second derivatives of delay for link i.

However, the delay of a wireless communication link between two neighbours is not only dependent on the traffic sending through the link but also on traffic in the vicinity due to wireless interference. Therefore, the single parameter form of the latency function $\ell_i(\bar{f}_i)$ does not suit wireless ad hoc networks. Correspondingly, the first and second derivative costs of paths must have different forms in wireless ad hoc networks, compared to the forms in wired networks. The problem for the derivative costs based on link model (4.1) is that they do not model the overall delay incurred by a flow to the network, which is needed for a routing algorithm to make routing decisions.

The key to the design of a wireless cost model is to divert from the traditional link model of network topologies and to view an ad hoc network as overlapping discs. Each of the discs represents a wireless transmission medium centred by a node. Given a medium access approach, a traffic flow sending by a node imposes delay to all flows within the disc. Therefore, we wish to establish a cost function over a disc, as opposed to the existing cost function over a link.

The service rate of a point-to-point link within a disc is determined by the wireless MAC approach used. As we have seen in the previous chapter, a joint optimization in wireless networks requires an optimal link scheduling at the MAC layer and a routing optimization at the network layer. Unfortunately, optimal link scheduling has been known to be very difficult and lacks sufficient results. In addition, any form of approximated optimal link scheduling requires a specialized MAC protocol. Instead, we focus on the design of a cost model based on a given MAC layer approach. It can be based on existing random access MAC layer approaches, such as the IEEE 802.11 protocol, as long as their behaviour is analytically modelled. Using the analytical model of medium access delay for each link, we derive the form of the cost function.

In this chapter, we organize the description of our design in a top-down structure and gradually fill the details. Firstly, we investigate the origination of delay in wireless transmission in section 4.2. Based on the observation, we propose a cost model in

section 4.3, called the wireless medium cost function. We understand that the wireless transmission is essentially a polling system and use a multi-class first-come-first-serve (FCFS) queue to model existing contention-based MAC protocol. Such a queueing model can be described by a M/G/1 queue. Therefore, with an analytical model of the service time of a given MAC protocol, such as the 802.11 DCF, the wireless medium cost is given. In appendix C, we present an example analytical model, where we use numerical approaches to find the closed form of 802.11 DCF service time.

In order to be used in an optimal routing algorithm, a necessary condition for a cost function is its convexity. In section 4.4, we discuss the convexity of the wireless medium cost and demonstrate that the convexity of the underlying MAC protocol service time is a sufficient condition. It is known that the 802.11 DCF exhibit a convex service time before entering a saturated state.

4.2 Where Does Delay of Wireless Communication Come From?

We anticipate that the rationale of optimization approaches in wired networks holds in wireless ad hoc networks: Routing decisions should take into account their overall impact on the network. The link cost information, including the first and second derivatives of delay, can be either computed from some analytical model or measured using the perturbation analysis. However, as we have discussed in the previous section, the point-to-point cost is not sufficient to model wireless cost because neighbouring transmissions may introduce costs to each other through interference. In this section, we will investigate the origination of delay in wireless transmission. Firstly, we will discuss components of communication delay from the view of a single flow. Then, we investigate how different communication flows may interfere with each other and incur delay mutually.

4.2.1 Components of Communication Delay

The overall delay experienced by all traffic in a network is the sum of delay experienced by each unit data at each node. The delay experienced at each node in wireless ad hoc networks includes two parts: the queueing delay and the service time. The queueing delay is the duration between the arrival of a unit data at an outgoing queue and the instant when it reaches the head of the queue. The service time is the time a unit data spent at the head of the queue until it is successfully received at its destination station. Therefore, the service time of a unit data includes: 1) the time it waits to access the medium - the medium access delay; 2) the duration to transmit a unit data under a given data rate - the transmission delay; 3) the time incurred by retransmission.

Due to the nature of ad hoc networks, we have neglected several components of delay. In ad hoc networks with single transmission channel, at most one packet can be successfully transmitted within the receiving range of any node. In contrast to wired networks, data transmission rather than the processing power of intermediate nodes is the bottleneck for the performance of end-to-end communications in ad hoc networks. Throughout this thesis, we do not consider the processing delay at intermediate nodes nor do we assume the existence of incoming queue at each node. Because the transmission range of wireless devices is trivial compared the speed of signal propagation, we neglect the propagation delay of data traffic.

In a network with substantial traffic demand, the queueing delay is the main contributor to the overall delay. Because nodes have finite queues, the long term data arrival rate is strictly smaller than the long term service rate[1]. However, within a certain duration, the data inter-arrival time may be shorter than the service time, which leads to queueing of subsequent data. Therefore, queueing delay is determined by the statistical distributions of inter-arrival time and the service time of data packets, including their mean, variance and probability distribution function (PDF), etc.

[1]Indeed, this is the capacity constraints enforced by routing algorithms

A common assumption for data arrivals in networking is that they are determined by a Poisson process. That is, the number of arrivals during any period is independent of the number of arrivals before the period. Namely, the data arrival is memoryless. While this assumption does not usually hold in real world, it is widely adopted in network modelling and analysis sfor several reasons:

1. Modelling of the real world traffic pattern, especially in the wireless ad hoc networks, is challenging and lacks of satisfying results [Floyd and Paxson, 2001].

2. Poisson arrival is reasonable approximation to persistent traffic, such as the TCP file transfer [Paxson and Floyd, 1995].

3. Poisson arrival eases the analysis of queueing model and of random access MAC algorithms behaviours. Strong results have been established.

It should be noted that a Possion generation of traffic at source nodes may not result in a Poisson arrival at intermediate nodes. We are aware of approaches that analyse the behaviour of queueing networks with traffic enters the networks according to a Poisson process, for example using parametric decomposition [Gross et al., 2008] or diffusing approximation [Bisnik and Abouzeid, 2009]. However, their results are approximations to the behaviours of wireless networks and lacks of quantifications on errors. Therefore, for simplicity reasons, we assume a Poisson arrival at all intermediate nodes. Given the Poisson arrival of data traffic, the long-term queueing delay is well analysed through a M/M/1 queueing model if the service time is exponential distributed or through a M/G/1 model is the service time is general distributed. The actual service time is determined by the traffic scenario and the MAC approach used.

In the next subsection, we discuss the impact on delay experienced by different traffic flows through wireless interference, which is independent to the assumption of Poisson traffic. In next the section, we will given the design of a wireless cost function, which is based on the assumption.

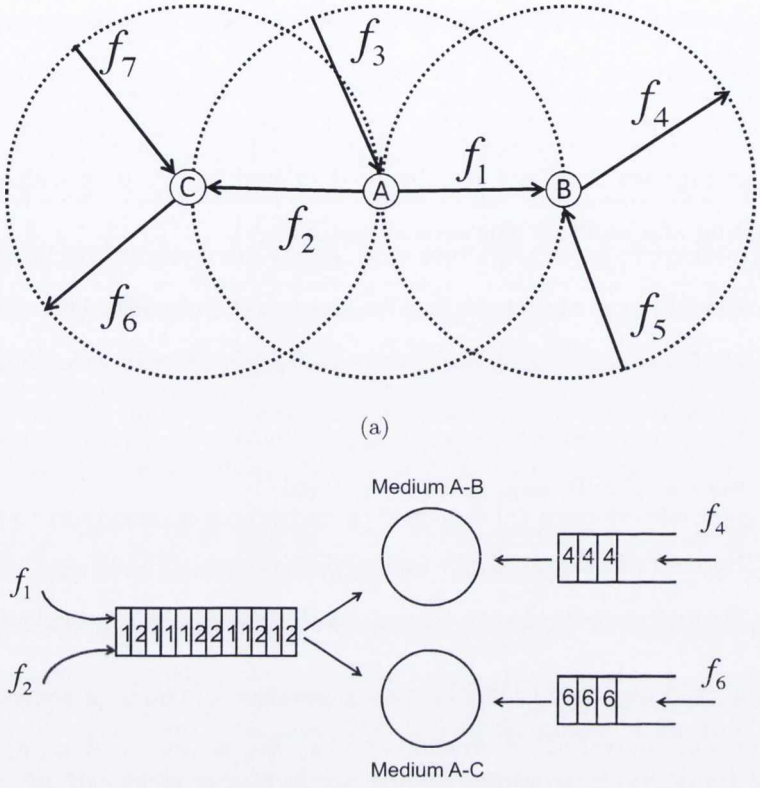


Fig. 4.1: A Demonstration of Wireless Interference

4.2.2 Wireless Interference

In the previous section, we have looked into a single flow at a single node and discussed the components of communication delay and the assumption on the distribution of the inter-arrival time of data traffic. In this subsection, we will discuss how different flows in a wireless network may interfere with each other and thus introduce delay mutually.

Let us firstly consider an example scenario as shown in Fig. 4.1(a). In this scenario, node A has two outgoing traffic flows to node B and C respectively and one incoming flow. Node B and node C each has two incoming flows and one outgoing flow. At each node, the outgoing traffic enters a FCFS queue waiting to access the medium, as shown

in Fig. 4.1(b). We assume all flows are one-hop and study their interference within the neighbourhood. We will extends this results to the interference between multi-hop communication flows in the next section.

Due to the broadcast nature of wireless transmission, outgoing flows of the same node share the same transmission medium. Therefore, even if they are addressed for different neighbours, they can not transmit at the same time[2]. This is different compared to wired network where a node can transmit through multiple links simultaneously. That means, all the outgoing flows of a node join the same FCFS queue of its transmitter. Therefore, outgoing flows of the same node interfere with each other in two ways: 1) They share the transmitter resource of the node. 2) They mutually affect their queueing time. In the literature, the interference among these flows are called *primary interference*. For example, in Fig. 4.1, flow f_1 and f_2 are primary interference of each other. Furthermore, we wish to emphasize that in a single FCFS queue, all flows have the same queueing time, despite their differences in service time. Therefore, the difference in delay experienced by f_1 and f_2 lies only in their medium access time.

In addition to the primary interference, the transmission of a data flow may also interfere with the transmissions/receptions of other nodes in vicinity. This type of interference can be described by the *protocol model* used in the literature [Gupta and Kumar, 2000, Jain et al., 2003]. In such a model, the coverage of the wireless signal of a node i is divided into a transmission range R_i and an interference range $\tilde{R}_i$, where $R_i \leq \tilde{R}_i$ in general. Without loss of generality, we assume the transmission range and the interference range are universal for all nodes in a network, i.e. $R_i = R$ and $\tilde{R}_i = \tilde{R}$ for all node i. When node i transmits to node j in its transmission range R, any other node k within its interference range $\tilde{R}$ can not successfully receive packets. And correspondingly, any node l whose interference range covers node j can not be transmitting. Extending the previous terminology, we call such an interference the *secondary interference*. Note

[2]This is not true if directional antennas are used or if multiplex techniques are used in omnidirectional antennas, such as the multiple-input and multiple-output (MIMO). However, since we are considering the case of single channel communication, we rule out these possibilities

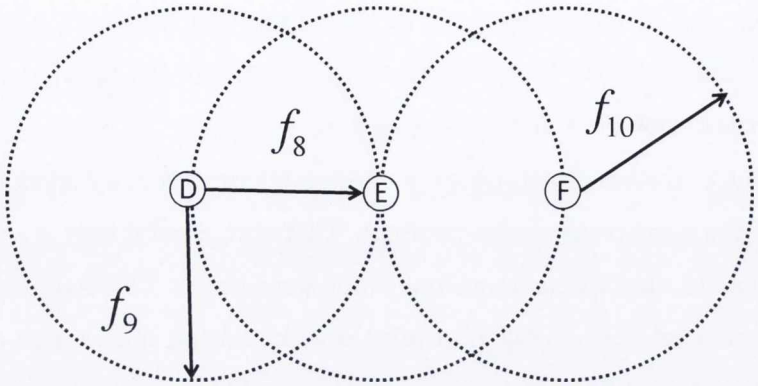


Fig. 4.2: Indirect interference: Flow f_9 and f_{10} do not share a queue nor do they share medium access. A change in either one will affect the delay experienced by the other.

that an equivalent presentation to the protocol model is the *physical interference model*, where the secondary interference is described using the notation of signal-to-noise ratio (SNR) [Jain et al., 2003].

In the example of Fig. 4.1, for simplicity reasons, we let $R = \tilde{R}$. In such a scenario, when f_1 is transmitting, its secondary interference includes two parts: 1) Any node within its interference range can not receive, i.e. flows f_3, f_5, f_7 have to wait; 2) Any node whose interference range covers node B can not transmit, i.e. flow f_4 has to wait.

It should be noted that this setting of secondary interference is optimistic, in that it only require the receiver to be free of interfering signals. In practise, the MAC approach may require that both transmitters and receivers are free of interference. For example, this is the case in IEEE 802.11 with the RTS/CTS mechanism. In such settings, flow f_6 has to wait while flow f_1 is ongoing.

So far, we have discussed the interference between flows with direct competition, either through a shared queue or through shared medium. We note that flows may interfere with each other even if they do not have direct contact. We term this type of interference as *indirect interference*.

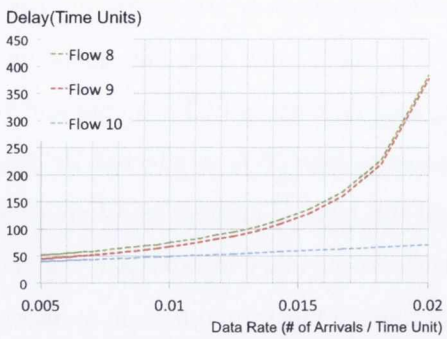
Let us consider an example shown in Fig. 4.2. Flow f_8 and f_9 are mutual primary

interference. Flow f_8 and f_{10} are mutual secondary interference. Because flow f_{10} prolongs the service time of flow f_8, it is easily seen that the queueing time of f_8 is affected by the traffic of f_{10}. Since f_8 and f_9 share a FCFS queue and have the same queueing time, we conclude that the queue time of f_9 is affected by f_{10}, despite that they do not have direct contact. We call this type of indirect interference as the *primary indirect interference*. It is associated with a secondary interference, e.g. between f_8 and f_{10}. Unlike direct interferences, it is not mutual but from one from to another, e.g. f_{10} to f_9. Finally, it leads to the same amount of queueing delay as the secondary interference.

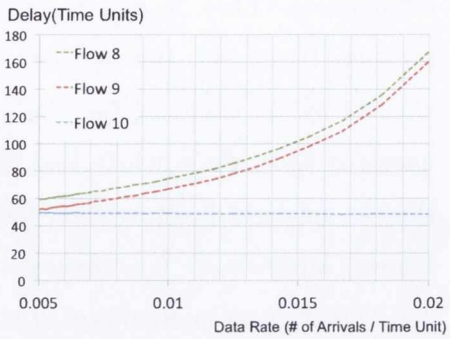
Recall that the queueing delay is determined by the statistical distribution of the inter-arrival time and the service time of data traffic. Therefore, a data flow may interfere with another flow by affecting the statistical distribution of its service time. We call this type of interference the *secondary indirect interference*. In the example of Fig. 4.2, the changes of flow f_9 may result in a different series of time instant that f_8 is at the head of the shared queue. To see this, consider two extreme cases: 1) If the amount of traffic for f_9 is zero. Then f_8 will request for medium access whenever its traffic arrives at the queue. 2) If the amount of traffic for f_9 is extremely large - within the service capacity though - then, the data units of f_8 is widely spread within the queue. Therefore, the time instant that f_8 requests for medium access is widely spread. The time instant that the medium is granted to f_{10} is thus different compared to case 1).

In the following, we will investigate the impact of the four types of interference. We set up the scenario shown in Fig. 4.2 via a simulator we have implemented[3]. In this scenario, we vary the data rate of one of the three flows and keep the other two flows constant rate. By doing so, we wish to demonstrate whether a certain type of interference is trivial or significant rather than quantify the amount of impact.

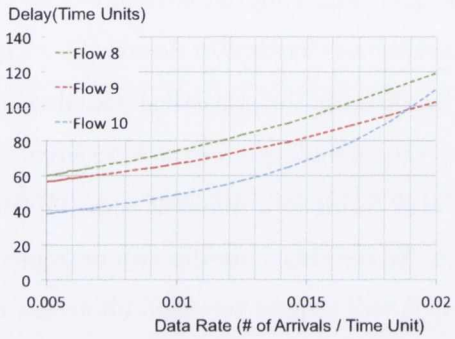
[3]We will describe the implementation of the simulator in chapter 6. Essentially, it is a M/M/1 queueing network with user-specific medium access latency.



(a) Delay Growth with an increasing f_8



(b) Delay Growth with an increasing f_9



(c) Delay Growth with an increasing f_{10}

Fig. 4.3: The impact of Different Interferences: Traffic arrival and service are determined by Poisson process. In each scenario, the expected inter-arrival time of one flow varies from 200 to 50 time units, while the other two are kept as 100 time units expected inter-arrival time. The expected service time for the medium is set as 25 time units

The result are illustrated in Fig. 4.3. As shown in Fig. 4.3(a), while flow f_8 is increasing, f_8 and f_9 experience a similar delay, which is increasing drastically. This demonstrates that the primary interference is a major source of cost incurred by traffic. This observation is confirmed by results shown in Fig. 4.3(b) - while f_9 increases, delay of both f_8 and f_9 grow in a similar way. Delay growth of f_{10} in Fig. 4.3(a) and of f_8 in Fig. 4.3(c) are caused by the secondary interference. Apparently, the scale of impact differs in

the two scenarios. Nonetheless, the secondary interference can not be ignored. As in Fig. 4.3(c), the cost growth experienced by f_9 shows that the primary indirect interference is also non-trivial. Finally, the delay reduction of flow f_{10} in Fig. 4.3(b) is due to the secondary indirect interference. The reduction is hardly observable. According to this example scenario, we choose to model the primary interference, the second interference and the primary indirect interference. We omit the secondary indirect interference in this thesis as it imposes trivial effect to the overall costs.

In summary, wireless interference presents a very different compared to wired networks. In this section, we have discussed the origination of delay in wireless communication. We demonstrated how a communication flow may interfere with the transmission of other flows by classifying four types of interferences: 1) primary interference; 2) secondary interference; 3) primary indirect interference; 4) secondary indirect interference. These interferences create couplings among multiple flows. In the next section, we give the design of a cost function that reflects such couplings.

4.3 The Wireless Medium Cost Model

In this section, we give the design of a cost function that models the delay experienced by all traffic flows in random access wireless ad hoc networks. Our goal is not to give an exact analytical function for a specific MAC protocol. But rather, we wish to model the interference between different flows so that the cost function can be used in an optimal routing algorithm, given the medium access delay of wireless links. In subsection 4.3.1, we will first give the queueing delay at each node with respect to all interfering flows. Then, based on the queueing model, we derive the overall cost incurred by a flow at each hop. In subsection 4.3.2, we will extend the function to model the cost of a path. The notations used in this section are shown in Table 4.1.

Notation	Meaning
f_p	1) Traffic flow along the path p for some communication, i.e. $p \in \mathcal{P}$ 2) Data rate of the traffic flow along the path p
$f_i^{(l)}$	1) The link traffic flow, i.e. $i \in \mathcal{L}$; 2) Data rate of the link i
$\mathcal{F}^{(l)}$	The set of link flow variables of a network.
$\tilde{C}(\cdot)$	The network cost function with respect to the set of link flows. It is a linear transformation of the original cost function, i.e. $\tilde{C}(\mathcal{F}^{(l)}) = C(\mathcal{F})$
$\mathcal{F}_a$	The set of outgoing link flows from node a
$\bar{f}_a$	The sum of flows of node a: $\bar{f}_a = \sum_{f_i^{(l)} \in \mathcal{F}_a} f_i^{(l)}$
$\mathcal{I}_{a,i}$	The set of flows that interferes with the medium access of flow $f_i^{(l)}$ at node a, i.e. its secondary interferences
$\mathcal{I}_a$	The set of flows that share medium access with any flow of node a: $\mathcal{I}_a = \cup_{i:f_i^{(l)} \in \mathcal{F}_a} \mathcal{I}_{a,i}$
$\mathcal{M}_{a,i}$	The sum of flows sharing the medium of $f_i^{(l)}$: $\mathcal{M}_{a,i} = f_i^{(l)} + \sum_{f_j^{(l)} \in \mathcal{I}_{a,i}} f_j^{(l)}$
$\pi_{a,i}$	The ordered pair of flow i and the sum of flows sharing its medium $\mathcal{M}_{a,i}$, i.e. $< f_i^{(l)}, \mathcal{M}_{a,i} >$
π_a	The set of $\pi_{a,i}$ of node a
$\ell_{a,i}^{(s)}(\cdot)$	Analytical function for the mean medium access delay experienced by $f_i^{(l)}$ at node a.
$\mathcal{S}_{a,i}^{(2)}(\cdot)$	Analytical function for the second moment of the medium access delay experienced by flow $f_i^{(l)}$
$\ell_a^{(q)}(\cdot)$	Queueing delay at node a
$\mathcal{N}_a^{(2)}$	The set of nodes within the distance $R + \tilde{R}$ of node a

Table 4.1: Notations in Section 4.3

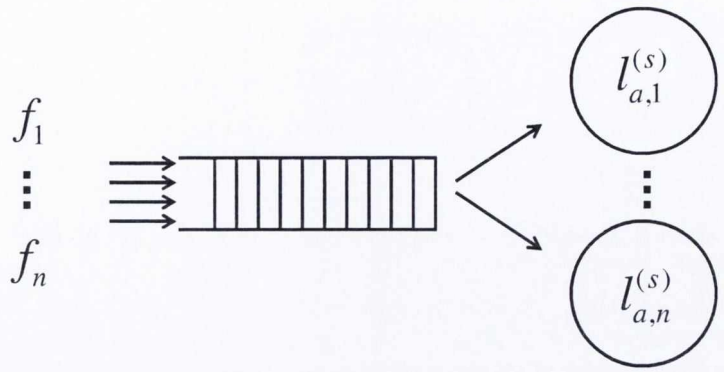


Fig. 4.4: Multi-class FCFS Queue: Traffic flows addressed for different neighbours have difference interferences hence experience different service time. The outgoing queue at each node can be modelled as a multi-class FCFS queue.

4.3.1 Wireless Medium Cost at Each Hop

Based on the observation we made in the previous section, we ignore the secondary indirect interference and consider the cost incurred by the primary interference, the secondary interference and the primary indirect interference. Since we assume a Poisson distributed inter-arrival time for all traffic, the mean data rate of a traffic flow is the only dominating factor for the cost it incurs. In this subsection, we give the cost incurred to the network as a function of data rates of flows sending out by a node.

4.3.1.1 Queueing Delay at Each Node

Firstly, we investigate the queueing delay. At each node, all outgoing traffic flows join the same FCFS queue. Since secondary interference takes place at the interference range of both the transmitter and the receiver. Flows that addressed for different neighbours may be interfered by different set of flows. They may experience different medium access delay. Therefore, the behaviour of traffic sending out by a node can be modelled as a multi-class FCFS queue: Each outgoing flow i at node a has an arrival rate, represented by $f_i^{(l)}$ - The superscript (l) is to distinguish from path flow and i is indexed within all

links of a network, i.e. $i \in \mathcal{L}$. Flow i also has a mean service time, denoted by $\ell_{a,i}^{(s)}$, which is the medium access delay it experiences at the node. We assume for now that the medium access delay is given by some analytical function of traffic flows sharing the medium access. That is, the medium access delay is given as $\ell_{a,i}^{(s)}(\mathcal{M}_{a,i})$, where $\mathcal{M}_{a,i} = f_i^{(l)} + \sum_{f_j^{(l)} \in \mathcal{I}_{a,i}} f_j^{(l)}$ is the sum of all flows that share the medium with $f_i^{(l)}$ at node a. Similarly, the second moment of the medium access delay is given by analytical function $\mathcal{S}_{a,i}^{(2)}(\mathcal{M}_{a,i})$

All flows in the FCFS queue experience the same queueing delay, denoted as $\ell_a^{(q)}$. To derive the queueing delay, we can treat the multi-class FCFS queue as a M/G/1 model [Kulkarni and Glazebrook, 2002]. This is because if each class of traffic has a independent memoryless Poisson arrival then the aggregated arrival of all flows is memoryless Poisson as well. Based on the delay analysis of M/G/1 queue, we have the following proposition:

Proposition 1. *Given the mean and the second moment of medium access for each flow, the queueing delay at each node is as follows:*

$$\ell_a^{(q)} = \frac{\sum_{i:f_i^{(l)} \in \mathcal{F}_a} \mathcal{S}_{a,i}^{(2)}(\mathcal{M}_{a,i}) f_i^{(l)}}{2\left(1 - \sum_{f_i^{(l)} \in \mathcal{F}_a} \ell_{a,i}^{(s)}(\mathcal{M}_{a,i}) f_i^{(l)}\right)} \tag{4.2}$$

The deduction of equation(4.2) is straightforward. Nonetheless, to our knowledge, the result is not explicitly stated in the literature. In appendix A, we will give a brief demonstration of the proof of the proposition.

Remark. We have the following remarks on Proposition 1

1) The queueing delay (4.2) can be seen as a function of outgoing flows and their secondarily interfering flows. Each outgoing flow corresponds to a set of secondarily interfering flows. Let $\pi_{a,i} =< f_i^{(l)}, \mathcal{M}_{a,i} >$ denote the ordered pair of an outgoing flow $f_i^{(l)} \in \mathcal{F}_a$ of node a and the sum of flows contending for medium access. Let $\pi_a = \{\pi_{a,i} \mid \forall i : f_i^{(l)} \in \mathcal{F}_a\}$ be the set of all the ordered pairs. Then, the left-hand side

of equation (4.2) is replaced by $\ell_a^{(q)}(\pi_a)$. This function give the impact on delay by each interfering flow. We will use the function form of queueing delay in the following design of medium cost.

2) We expect that the mean and the second moment of medium access delay - $\ell_{a,i}^{(s)}(\cdot)$ and $\mathcal{S}_{a,i}^{(2)}(\cdot)$ to be given by studies of the MAC layer approaches used in an ad hoc network. We will show a case study based on an analytical model of the IEEE 802.11 protocol [Malone et al., 2007] in Appendix C.

3) For the sake of simplicity, we define $\ell_{a,i}^{(s)}(\cdot)$ and $\mathcal{S}_{a,i}^{(2)}(\cdot)$ as univariate functions. That is, they take in the sum of flows accessing the medium $\mathcal{M}_{a,i}$. We acknowledge that this may not always be case for analytical functions of MAC approaches. A more general form for each of the analytical functions is to take interfering flows separately. In this case, instead of the sum of flows, we let $\mathcal{M}_{a,i}$ denotes the set of flows accessing the medium, i.e. $\mathcal{M}_{a,i} = \mathcal{I}_{a,i} \cup \{f_i^{(l)}\}$ Equation(4.2) remains the same.

4.3.1.2 Costs Incurred by A Flow

The delay that flow $f_i^{(l)} \in \mathcal{F}_a$ experiences at the node a, denoted as $\ell_{a,i}$, is the sum of queueing delay and medium access delay. The cost of flow $f_i^{(l)}$ at node a is given as:

$$\ell_{a,i}(\pi_a) f_i^{(l)} = \left(\ell_{a,i}^{(s)}(\mathcal{M}_{a,i}) + \ell_a^{(q)}(\pi_a)\right) f_i^{(l)} = \left(\ell_{a,i}^{(s)}(\mathcal{M}_{a,i}) + \frac{\sum_{j:f_j^{(l)} \in \mathcal{F}_a} \mathcal{S}_{a,j}^{(2)}(\mathcal{M}_{a,j}) f_j^{(l)}}{2\left(1 - \sum_{j:f_j^{(l)} \in \mathcal{F}_a} \ell_{a,j}^{(s)}(\mathcal{M}_{a,j}) f_j^{(l)}\right)}\right) f_i^{(l)} \tag{4.3}$$

Indeed, this expression of cost is factored by the three major types of interference as we have observed in Section 4.2.2:

1. Each $f_j^{(l)}$ such that $f_j^{(l)} \in \mathcal{F}_a$ and $j \neq i$ is a primary interference with flow $f_i^{(l)}$;
2. Each $f_r^{(l)} \in \mathcal{I}_{a,i}$ is part of $\mathcal{M}_{a,i}$ and is a secondary interference with flow $f_i^{(l)}$;
3. Each $f_r^{(l)} \in \mathcal{I}_{a,j}$ such that $f_j^{(l)} \in \mathcal{F}_a$ and $j \neq i$ is part of $\mathcal{M}_{a,j}$ and is a primary indirect interference of flow $f_i^{(l)}$

Conversely, flow $f_i^{(l)}$ is a factor of costs of flows in the vicinity. In the following, We define the wireless medium cost of node a as the overall cost in which its outgoing flows play a factor. To ease the formal definition, let $\mathcal{N}_a^{(2)}$ denote the set of two-hop interfering nodes of node a, namely $\mathcal{N}_a^{(2)} = \{b \in \mathcal{N} | \text{distance between } a \text{ and } b \text{ is less than } R + \tilde{R}\}$. All nodes that are interfered by node a are in the set of $\mathcal{N}_a^{(2)}$, although not vice versa. To distinguish the interfering nodes in the set of $\mathcal{N}_a^{(2)}$, for any node b, let $\mathcal{I}_b = \cup_{j:f_j^{(l)} \in \mathcal{F}_b} \mathcal{I}_{b,j}$ denote the set of flows that share medium access with any flow of node b. Then node b is interfered by node a if some flow in $\mathcal{I}_b$ belongs to node a, namely $\mathcal{F}_a \cap \mathcal{I}_b \neq \emptyset$.

Now, we are in the position to give a formal definition of the proposed wireless medium cost:

Definition 1. The wireless medium cost of node a is a function of the set of its outgoing traffic, defined as:

$$C_{\&,a}(\mathcal{F}_a) = \sum_{i:f_i \in \mathcal{F}_a} \ell_{a,i}(\pi_i) f_i^{(l)} + \sum_{b:b \in \mathcal{N}_a^{(2)}} \sum_{\substack{r:f_r^{(l)} \in \mathcal{F}_b \\ \mathcal{F}_a \cap \mathcal{I}_{b,r} \neq \emptyset}} \ell_{b,r}(\pi_r) f_r^{(l)}$$

$$= \ell_a^{(q)}(\pi_a)\bar{f}_a + \sum_{i:f_i^{(l)} \in \mathcal{F}_a} \ell_{a,i}^{(s)}(\mathcal{M}_{a,i}) f_i^{(l)} + \sum_{\substack{b:b \in \mathcal{N}_a^{(2)} \\ \mathcal{F}_a \cap \mathcal{I}_b \neq \emptyset}} \left(\ell_b^{(q)}(\pi_b)\bar{f}_b + \sum_{\substack{r:f_r^{(l)} \in \mathcal{F}_b \\ \mathcal{F}_a \cap \mathcal{I}_{b,r} \neq \emptyset}} \ell_{b,r}^{(s)}(\mathcal{M}_{b,r}) f_r^{(l)} \right) \quad (4.4a)$$

The wireless medium cost give all potential costs that are affected by the traffic of a node, through the three major types of interferences. In a network with substantial traffic demand, the queueing delay is the dominating factor of cost, e.g. it may be one or several orders of magnitude larger than the medium access delay. In such a case, the wireless medium cost can be reduced to

$$C_{\&,a}(\mathcal{F}_a) \approx \ell_a^{(q)}(\pi_a)\bar{f}_a + \sum_{\substack{b:b \in \mathcal{N}_a^{(2)} \\ \mathcal{F}_a \cap \mathcal{I}_b \neq \emptyset}} \ell_b^{(q)}(\pi_b)\bar{f}_b \quad (4.4b)$$

Now, we deduce the derivatives of the wireless medium cost at each node a with respect to an outgoing flow $f_i^{(l)} \in \mathcal{F}_a$ of node a. To do so, we start from several observations with respect to the components of the derivative.

Firstly, for any node x and any flow $f_i^{(l)}$, the derivative $\frac{\partial \mathcal{M}_{x,j}}{\partial f_i^{(l)}} = \frac{\mathrm{d}\, \mathcal{M}_{x,j}}{\mathrm{d}\, f_i^{(l)}} = 0$, if either of the following statements holds: 1) $f_i^{(l)} \in \mathcal{F}_x \wedge i \neq j$; 2) $f_i^{(l)} \notin \mathcal{F}_x \wedge f_i^{(l)} \notin \mathcal{I}_{x,j}$. Conversely, the derivative $\frac{\partial \mathcal{M}_{x,j}}{\partial f_i^{(l)}} = \frac{\mathrm{d}\, \mathcal{M}_{x,j}}{\mathrm{d}\, f_i^{(l)}} = 1$, if either of the following statements holds: 1) $f_i^{(l)} \in \mathcal{F}_x \wedge i = j$; 2) $f_i^{(l)} \notin \mathcal{F}_x \wedge f_i^{(l)} \in \mathcal{I}_{x,j}$

Secondly, let $\pi_{x,j}(1)$ and $\pi_{x,j}(2)$ denote the first and the second elements of the ordered pair $\pi_{x,j}$ [4]. The Jacobian of the queueing delay at node x with respect to j^{th} link is given as $\frac{\partial \ell_x^{(q)}(\pi_x)}{\partial \pi_{x,j}} = \{\frac{\partial \ell_x^{(q)}(\pi_x)}{\partial \pi_{x,j}(1)}, \frac{\partial \ell_x^{(q)}(\pi_x)}{\partial \pi_{x,j}(2)}\}$. It is easily known that if $f_j^{(l)} \notin \mathcal{F}_x$, then $\frac{\partial \ell_x^{(q)}(\pi_x)}{\partial \pi_{x,j}(1)} = \frac{\partial \ell_x^{(q)}(\pi_x)}{\partial \pi_{x,j}(2)} = 0$. For $f_j \in \mathcal{F}_x$, we have:

$$\begin{cases} \frac{\partial \ell_x^{(q)}(\pi_x)}{\partial \pi_{x,j}(1)} = \frac{\mathcal{S}_{x,j}^{(2)}(\mathcal{M}_{x,j})B + \ell_{x,j}^{(s)}(\mathcal{M}_{x,j})A}{2B^2} \\ \frac{\partial \ell_x^{(q)}(\pi_x)}{\partial \pi_{x,j}(2)} = \frac{\mathcal{S}_{x,j}^{(2)'}(\mathcal{M}_{x,j}) f_j^{(l)} B + \ell_{x,j}^{(s)'}(\mathcal{M}_{x,j}) f_j^{(l)} A}{2B^2} \end{cases} \tag{4.5}$$

where $A = \sum_{r:f_r^{(l)} \in \mathcal{F}_x} \mathcal{S}_{x,r}^{(2)}(\mathcal{M}_{x,r}) f_r^{(l)}$ and $B = 1 - \sum_{f_r^{(l)} \in \mathcal{F}_x} \ell_{x,r}^{(s)}(\mathcal{M}_{x,r}) f_r^{(l)}$. And, $\mathcal{S}_{x,j}^{(2)'}(\cdot)$ and $\ell_{x,j}^{(s)'}(\cdot)$ are first derivatives of their corresponding functions. According to the chain rule of partial derivatives, the partial derivative of queueing delay at any node x with respect to a flow i is as follows:

$$\frac{\partial \ell_x^{(q)}(\pi_x)}{\partial f_i^{(l)}} = \sum_{j:f_j^{(l)} \in \mathcal{F}_x} \frac{\partial \ell_x^{(q)}(\pi_x)}{\partial \pi_{x,j}} \frac{\partial \pi_{x,j}}{\partial f_i^{(l)}} = \sum_{j:f_j^{(l)} \in \mathcal{F}_x} \Big(\frac{\partial \ell_x^{(q)}(\pi_x)}{\partial \pi_{x,j}(1)} \frac{\partial f_j^{(l)}}{\partial f_i^{(l)}} + \frac{\partial \ell_x^{(q)}(\pi_x)}{\partial \pi_{x,j}(2)} \frac{\partial \mathcal{M}_{x,j}}{\partial f_i^{(l)}} \Big) \tag{4.6}$$

Finally, we give the partial derivative of the wireless medium cost at node a with respect to outgoing flow $f_i^{(l)} \in \mathcal{F}_a$.

$$\frac{\partial C_{\&,a}(\mathcal{F}_a)}{\partial f_i^{(l)}} = \sum_{j:f_j \in \mathcal{F}_a} \frac{\partial [\ell_{a,j}(\pi_j) f_j^{(l)}]}{\partial f_i^{(l)}} + \sum_{b:b \in \mathcal{N}_a} \sum_{\substack{r:f_r^{(l)} \in \mathcal{F}_b \\ \mathcal{F}_a \cap \mathcal{I}_{b,r} \neq \emptyset}} \frac{\partial [\ell_{b,r}(\pi_r) f_r^{(l)}]}{\partial f_i^{(l)}} \tag{4.7a}$$

[4] $\pi_{x,j}(1)$ and $\pi_{x,j}(2)$ are indeed $f_j^{(l)} \in \mathcal{F}_x$ and $\mathcal{M}_{x,j}$ respectively. We create this notations in order to avoid confusions between partial derivatives with respect to each elements separately and partial derivatives with respect to link flows

Combining equations(4.3), (4.5), (4.6) into (4.7a), a more detail version of the wireless medium cost is as follows:

$$\begin{aligned}
&\frac{\partial C_{\&,a}(\mathcal{F}_a)}{\partial f_i^{(l)}} \\
=&\frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,i}(1)} \bar{f}_a + \frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,i}(2)} \bar{f}_a + \ell_a^{(q)}(\pi_a) + \ell_{a,i}^{(s)'}(\mathcal{M}_{a,i}) f_i^{(l)} + \ell_{a,i}^{(s)}(\mathcal{M}_{a,i}) \\
+ &\sum_{\substack{b:b\in\mathcal{N}_a^{(2)} \\ f_i^{(l)}\in\mathcal{I}_b}} \sum_{\substack{j:f_j^{(l)}\in\mathcal{F}_b \\ f_i^{(l)}\in\mathcal{I}_{b,j}}} \left(\frac{\partial \ell_b^{(q)}(\pi_b)}{\partial \pi_{b,j}(2)} \bar{f}_b + \ell_{b,j}^{(s)'}(\mathcal{M}_{b,j}) f_j^{(l)} \right)
\end{aligned} \tag{4.7b}$$

If we omit the medium access delay and focus on only the queueing delay as suggested by equation (4.4b), the wireless medium cost reduces to a neat form as follows:

$$\frac{\partial C_{\&,a}(\mathcal{F}_a)}{\partial f_i^{(l)}} \approx \overbrace{\frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,i}(1)} \bar{f}_a + \frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,i}(2)} \bar{f}_a + \ell_a^{(q)}(\pi_a)}^{\text{Marginal cost incurred by } f_i^{(l)} \text{ at node } a} + \overbrace{\sum_{\substack{b:b\in\mathcal{N}_a^{(2)} \\ f_i^{(l)}\in\mathcal{I}_b}} \frac{\partial \ell_b^{(q)}(\pi_b)}{\partial \pi_{b,j}(2)} \bar{f}_b}^{\text{Marginal cost incurred by } f_i^{(l)} \text{ at a's two-hop neighbours}} \tag{4.7c}$$

This completes our investigation of wireless medium cost with respect to the outgoing traffic of a node, which models the overall cost incurred by the flow at the node. It should be noted that the sum of wireless medium costs of all nodes in a network is strictly larger than the sum of costs in general, except for some extreme topology where the two sums may equal. The point of the wireless medium cost is that it models the impact of nodes' actions. More specifically, we have the following observation of the wireless medium cost:

$$\forall i, a : i \in \mathcal{L},\ f_i^{(l)} \in \mathcal{F}_a, \quad \frac{\partial C(\mathcal{F})}{\partial f_i^{(l)}} = \frac{\partial C_{\&,a}}{\partial f_i^{(l)}} \tag{4.8}$$

In next subsection, we extend the results of wireless medium cost to the path flows, which can then be used by optimal routing algorithms.

4.3.2 Wireless Medium Cost of a Path

Based on the wireless medium cost at each node of a network, we now define the wireless medium cost of a path.

Definition 2. The wireless medium cost of path $p \in \mathcal{P}_w \subseteq \mathcal{P}$ is a function of the set of its outgoing traffic, defined as the sum of the wireless medium cost at each hop:

$$C_{\&,p}(f_p) = \sum_{a \in p} C_{\&,a}(\mathcal{F}_a) \tag{4.9}$$

Accordingly, the derivative of the wireless medium cost of path p with respect to the path flow f_p is given as:

$$C'_{\&,p}(f_p) = \frac{\partial C_{\&,p}(f_p)}{\partial f_p} = \sum_{\substack{a \in p \\ f_i^{(l)} \in \mathcal{F}_a}} \frac{\partial C_{\&,a}(\mathcal{F}_a)}{\partial f_i} \frac{\partial f_i^{(l)}}{\partial f_p} = \sum_{\substack{a \in p, f_i^{(l)} \in \mathcal{F}_a \\ i \in \mathcal{L}_p}} \frac{\partial C_{\&,a}}{\partial f_i^{(l)}} \tag{4.10}$$

where $\mathcal{L}_p$ represents the set of links used by path p. The last equality above is established based on the following observations: If $i \notin \mathcal{L}_p$, then $\frac{\partial f_i^{(l)}}{\partial f_p} = 0$. If we assume all paths established by a routing algorithm are loop-free, then for $i \in \mathcal{L}_p$, we have $\frac{\partial f_i^{(l)}}{\partial f_p} = 1$. Equation (4.10) states that the marginal medium cost of a path is the sum of marginal medium cost at each hop. It is easily verified, according to equation (4.8) and (4.10), the marginal medium cost of a path is the marginal cost of the network with respect to the path flow, i.e.

$$C'_{\&,p}(f_p) = \frac{\partial C(\mathcal{F})}{\partial f_p}$$

Therefore, through the concept of the wireless medium cost, we established an equivalence of equation (4.1a) for wireless ad hoc networks, namely (4.10)

So far, we have defined the wireless medium cost for each hop and each path. We have also given their first derivatives. Let us assume for now that the wireless medium cost is strictly convex, which we will discuss in Section (4.4). Under the convexity condition, these results are sufficient for first-order routing algorithms to optimize wireless ad hoc routing problems. However, in order to use second-order optimization methods, we need to find the second derivative of the overall network cost with respect to each path flow, which indeed can also be achieved through the use of the wireless medium cost.

For any path flow f_p and f_q, the second derivative cost of the network is given

$$\frac{\partial^2 C(\mathcal{F})}{\partial f_p \partial f_q} = \frac{\partial}{\partial f_q}\left(\frac{\partial C(\mathcal{F})}{\partial f_p}\right) = \frac{\partial}{\partial f_q}\left(\frac{\partial C_{\&,p}(f_p)}{\partial f_p}\right)$$
$$= \sum_{\substack{a \in p, f_i^{(l)} \in \mathcal{F}_a \\ i \in \mathcal{L}_p}} \frac{\partial}{\partial f_q}\left(\frac{\partial C_{\&,a}(\mathcal{F}_a)}{\partial f_i^{(l)}}\right) = \sum_{\substack{b \in q, f_j^{(l)} \in \mathcal{F}_b \\ j \in \mathcal{L}_q}} \frac{\partial}{\partial f_p}\left(\frac{\partial C_{\&,b}(\mathcal{F}_b)}{\partial f_j^{(l)}}\right) \tag{4.11}$$

The last two steps are partial differentiations of marginal medium costs at each hop - namely $\frac{\partial C_{\&,a}(\mathcal{F}_a)}{\partial f_i^{(l)}}$ and $\frac{\partial C_{\&,b}(\mathcal{F}_b)}{\partial f_j^{(l)}}$ - with respect to path flows f_q and f_p. We omit the cumbersome yet standard expansion of the differentiations. Nonetheless, it is worth noting the difference in differentiations with respect to path flow compared to differentiations with respect to link flow: As we have described previously, the value $\frac{\partial \mathcal{M}_{x,j}}{\partial f_i^{(l)}}$ is either 0 or 1; The value $\frac{\partial \mathcal{M}_{x,j}}{\partial f_p} = \sum_{i:i \in \mathcal{I}_{x,j,p}} 1$, where $\mathcal{I}_{x,j,p} = \{i | f_i^{(l)} \in \mathcal{L}_p \wedge f_i^{(l)} \in \mathcal{I}_{x,j} \cup \{f_j^{(l)}\}\}$, can be any non-negative integer.

In this section, we have analyzed the queueing model for wireless ad hoc networks. Based on the queueing model, we propose a cost function, called the wireless medium cost, that models the delay cost incurred by a traffic flow to the overall network. We start from wireless medium at each hop and extends the results to each path. Although wireless medium costs of each path or each node do not sum to the overall cost of a network, we have demonstrated that the first and the second derivatives of the wireless medium costs give the derivatives of the network costs with respect to path flows. The wireless medium cost is an analytical model of costs incurred due to wireless interferences. It encapsulates the complexity of wireless interferences and MAC approaches into a function that can be used by optimal routing algorithms.

4.4 Convexity of Wireless Routing Optimization Problems

We can design an optimal routing algorithm for wireless ad hoc networks based on the first and the second derivatives of wireless medium costs. However, a prerequisite for

existing convex optimization approaches to converge is the convexity of the objective function.

Existing studies of optimal routing approaches have rarely shown the verification of convexity for routing optimization problems. This is mainly because that the verification of convexity in wired networks is trivial, assuming that the link delay is convex. More specifically, the objective cost function in wired networks can be transformed as follows:

$$C(\mathcal{F}) = C(\mathcal{R}^{-1}\mathcal{F}^{(l)}) = \tilde{C}(\mathcal{F}^{(l)}) = \sum_{i \in \mathcal{L}} \ell_i(f_i^{(l)}) f_i^{(l)} \tag{4.12}$$

where $\mathcal{F}^{(l)}$ is a linear transformation of path flows $\mathcal{F}$ and represents the set of link flows in a network. $\tilde{C}(\cdot) = C(\cdot) \circ \mathcal{R}^{-1}$ is a composition function that gives network costs given the set of link flows. Based on the affine-invariant property of convex functions, in order to verify the convexity of $C(\mathcal{F})$, one needs only to show the convexity of $\ell_i(f_i^{(l)}) f_i^{(l)}$, which stands strictly if $\ell_i(f_i^{(l)})$ is convex.

However, the convexity can not be easily established in wireless ad hoc networks, even if the medium delay function is convex. As we have demonstrated, the latency function $\ell(\cdot)$ in wireless communications is subject to a set of mutually interfered link flows, which complicates and invalidates the last step in equation (4.12). In this section, we give an analysis of the convexity in wireless delay-based cost function.

Following the same rationale as equation (4.12), we transform the objective cost function into link rate based cost function $\tilde{C}(\mathcal{F}^{(l)})$. For the purpose of convenience, we divide the cost function into the summation of medium access delay and queueing delay

and prove their convexity separately.

$$C(\mathcal{F}) = \sum_{a\in\mathcal{N}} \ell_a^{(q)}(\pi_a)\bar{f}_a + \sum_{\substack{i\in\mathcal{L} \\ a:f_i\in\mathcal{F}^{(l)}}} \ell_{a,i}^{(s)}(\mathcal{M}_{a,i})f_i^{(l)}$$

$$= \sum_{a\in\mathcal{N}} \tilde{l}_a^{(q)}(\tilde{\mathcal{F}}_a^{(l)})T_a\tilde{\mathcal{F}}_a^{(l)} + \sum_{\substack{i\in\mathcal{L} \\ a:f_i\in\mathcal{F}^{(l)}}} \tilde{l}_{a,i}^{(s)}(\tilde{\mathcal{F}}_a^{(l)})T_{a,i}\tilde{\mathcal{F}}_a^{(l)} \tag{4.13a}$$

$$= \sum_{a\in\mathcal{N}} \tilde{C}_a^{(q)}(\tilde{\mathcal{F}}_a^{(l)}) + \sum_{\substack{i\in\mathcal{L} \\ a:f_i\in\mathcal{F}^{(l)}}} \tilde{C}_{a,i}^{(s)}(\tilde{\mathcal{F}}_a^{(l)}) \tag{4.13b}$$

$$= \tilde{C}(\mathcal{F}^{(l)}) \tag{4.13c}$$

The second equality (4.13a) is because that $\mathcal{M}_{a,i}, \pi_a$, $\bar{f}_a$ and $f_i^{(l)}$ can all be seen as a result of linear transformation of the link flow vector $\tilde{\mathcal{F}}_a^{(l)} = \mathcal{F}_a \cup \mathcal{I}_a$. We use $\tilde{\ell}_a^{(q)}(\cdot)$ and $\tilde{\ell}_{a,i}^{(s)}(\cdot)$ to represent the linearly transformed delay functions taking $\tilde{\mathcal{F}}_a^{(l)}$ as arguments. T_a and $T_{a,i}$ are linear transformations such that $\bar{f}_a = T_a\tilde{\mathcal{F}}_a^{(l)}$, $f_i^{(l)} = T_{a,i}\tilde{\mathcal{F}}_a^{(l)}$. Therefore, as shown in the third equality (4.13b), the network cost can written as a summation of functions of $\tilde{\mathcal{F}}_a^{(l)}$. Once more, $\tilde{\mathcal{F}}_a^{(l)}$ is a subset of the network link flows $\mathcal{F}^{(l)}$, which means it can be linearly transformed from $\mathcal{F}^{(l)}$. as shown in the final equality (4.13c). According to the affine-invariant property of convexity, in order to demonstrate the convexity of $C(\mathcal{F})$, we need only to demonstrate the convexity of each $\tilde{C}_a^{(q)}(\mathcal{F}^{(l)})$ and $\tilde{C}_{a,i}^{(s)}(\mathcal{F}^{(l)})$.

Firstly, we show that the queueing delay at each node is indeed convex under certain condition:

Lemma 1. *If the following condition stands - for each node $a \in \mathcal{N}$ and each link $f_i^{(l)} \in \mathcal{F}_a$ the medium access delay $\ell_{a,i}^{(s)}(\cdot)$ and the second moment of medium access delay $\mathcal{S}_{a,i}^{(2)}(\cdot)$ are convex, then the queueing delay at each node $\ell_a^{(q)}(\pi_a) = \tilde{\ell}_a^{(q)}(\tilde{\mathcal{F}}_a^{(l)})$ is strictly convex over vector $\tilde{\mathcal{F}}_a^{(l)}$.*

Proof. The main rationale is to demonstrate that the second partial derivative of queueing delay with respect to $f_i^{(l)}$ and $f_j^{(l)} \in \tilde{\mathcal{F}}_a^{(l)} = \mathcal{F}_a \cup \mathcal{I}_a$ is strictly positive. For detailed proof, see Appendix B.1 □

Lemma 2. *If the queueing delay $\tilde{\ell}_a^{(q)}(\tilde{\mathcal{F}}_a^{(l)})$ is (strictly) convex, then the queueing cost at each node $\tilde{C}_a^{(q)}(\tilde{\mathcal{F}}_a^{(l)}) = \tilde{\ell}_a^{(q)}(\tilde{\mathcal{F}}_a^{(l)})T_a\tilde{\mathcal{F}}_a^{(l)}$ is (strictly) convex w.r.t. the link flow set $\tilde{\ell}_a^{(q)}(\tilde{\mathcal{F}}_a^{(l)})$. Similarly, if the medium access delay $\tilde{l}_{a,i}^{(s)}(\tilde{\mathcal{F}}_a^{(l)})$ is (strictly) convex, then the medium access delay cost $\tilde{C}_{a,i}^{(s)}(\tilde{\mathcal{F}}_a^{(l)}) = \tilde{l}_{a,i}^{(s)}(\tilde{\mathcal{F}}_a^{(l)})T_{a,i}\tilde{\mathcal{F}}_a^{(l)}$ at each link is (strictly) convex.*

Proof. The proof is based on the definition of convex functions. For detailed proof, see Appendix B.2. □

With the results of Lemma 1 and 2, the convexity of the objective cost function in wireless networks are straightforward. In summary, we have the following proposition,

Proposition 2. *If the medium access delay $\ell_{a,j}^{(s)}(\cdot)$ and the second moment of medium access delay are both convex, then the objective function $C(\mathcal{F})$ is convex.*

Proof. See Appendix B.3. □

With the convexity of wireless delay costs established, an immediate result is that, we can use convex optimization approaches on the wireless medium cost function to achieve network optimization. Formally speaking:

Corollary 1. *There exists a network routing solution $\mathcal{F}^*$ such that*

$$C(\mathcal{F}^*) = \min\{C(\mathcal{F})|\forall\mathcal{F} \in \textbf{\textit{dom}}\ C(\cdot)\}$$

Or equivalently,

There exists a solution $\mathcal{F}^*$ that solves the wireless network optimization problem (1.2).

Using the KKT condition [Boyd and Vandenberghe, 2004], another derivative result of the Proposition 2 is the optimal Wardrop equilibrium for wireless ad hoc networks using the marginal wireless medium costs of each paths.

Corollary 2. *For each communication w, given its traffic demand $\mathcal{T}_w$, all its available paths $\{\mathcal{P}_w\}$ can be numbered $1, ...N_{use}, ...N_{all}$ so that:*

$$C'_{\&,1}(f_1) = ... = C'_{\&,N_{use}}(f_{N_{use}}) = M \leq C'_{\&,N_{use}+1}(f_{N_{use}}) \leq ... \leq C'_{\&,N_{all}}(f_{N_{all}})$$

$$f_i > 0, \quad i = 1, ..., N_{use}$$

$$f_i = 0, \quad i = N_{use} + 1, ..., N_{all}$$

Then the network solution is optimal, i.e. $\mathcal{F} = \mathcal{F}^*$.

Similar to the case of routing problems in wired networks, the optimal Wardrop equilibrium is an necessary and sufficient condition for network optimality. Therefore, a simplistic routing algorithm can given by balancing marginal costs at each source node.

This completes our verification of convexity for wireless optimal routing problems. In the next section, we summarize the information needed to compute wireless medium costs and describe the mechanism to retrieve the information [Roughgarden and Tardos, 2002].

4.5 Summary

In this chapter, we have given the definition of the wireless medium cost, which describes the overall costs incurred of a traffic flow to its neighbours. Therefore, the derivatives of wireless medium costs coincide with the derivatives of the network costs with respect to a traffic flow. We have demonstrated that the wireless medium cost function is strictly convex. In the following chapter, we devise an optimal routing algorithm using the wireless medium costs.

Chapter 5

Distributed Optimal Routing

In the previous chapter, we have described the design of the wireless medium cost and verified the convexity of wireless costs. Using the first and the second derivatives of the wireless medium cost, existing optimal routing approaches can function in wireless ad hoc networks. However, their performance are likely to suffer. We have summarized in chapter 2 and 3 a number of open questions for optimal routing in wireless networks, regarding to route discovery, convergence rate and distributed manner. In this chapter, we present an optimal routing algorithm that aims to address these issues.

5.1 Overview

In this section, we firstly reiterate the design objective of our wireless optimal routing algorithm. Then, we describe the organization of this chapter.

Our formulation of the wireless optimal routing is given in system (1.2). To address this problem, we base our approach on Newton's method for its superior convergence rate compared to first-order methods. We address the constraints using available tools reviewed previously. We aim to design a node-level distributed optimal routing algorithm. According to the discussion in section 3.8.2, we will use the path flow as handle variables to avoid the creation of large amount of subproblems.

Notation	Meaning
$\hat{\ell}^{(s)}$	Medium access delay at saturation. It is the upper bound of $\ell_{a,i}^{(s)}(\cdot)$
$\mathcal{Q}_p^a$	Quota assigned to flow f_p at node a, which is the portion of available capacity of node a. The value is set as infinity if f_p does not use the queue or the medium of node a, i.e. $f_p \notin \mathcal{F}_a \cup \mathcal{I}_a$
$\mathcal{Q}^a$	The set of quota assigned by node a to all flows that use its queue or medium access resources, namely, $\mathcal{Q}^a = \{\mathcal{Q}_p^a\}_{f_p \in \mathcal{F}_a \cup \mathcal{I}_a}$
$\mathcal{Q}_p$	The aggregated quota assigned to flow f_p, namely $\mathcal{Q}_p = \min\{\mathcal{Q}_p^a \| a \in \mathcal{N}\}$
$\mathcal{Q}$	The set of quotas for all flows. $\mathcal{Q} = \{\mathcal{Q}_p \| f_p \in \mathcal{F}\} = \{\mathcal{Q}_p \| p \in \mathcal{P}\}$
$\kappa(a)$	The set of indices i of capacity constraints $g_i(\mathcal{F}) < 0$, such that $g_i(\mathcal{F})$ represents either the queueing capacity constraints at node a or the medium capacity constraints of outgoing links of node a

Table 5.1: Notations in Chapter 5

The rest of the chapter is organized as follows. In section 5.2, we discuss our design that addresses the constraints and decomposes the couplings introduced by constraints down to flow level. In section 5.3, we describe our approach to decouple the objective functions. Next, we transform our algorithm from a flow-level distributed to a node-level distributed algorithm. In section 5.4.1, we give the design of route discovery and maintenance for optimal routing algorithms. The structure of our routing algorithm is illustrated in Fig. 5.1

5.2 Quota-based Interior-Point Method

In this section, we address the constraints in wireless routing optimization problem. This involves two parts: in subsection 5.2.1, we describe our choice on addressing constraints in a centralized approach, that is to keep the converging sequence of solutions feasible at

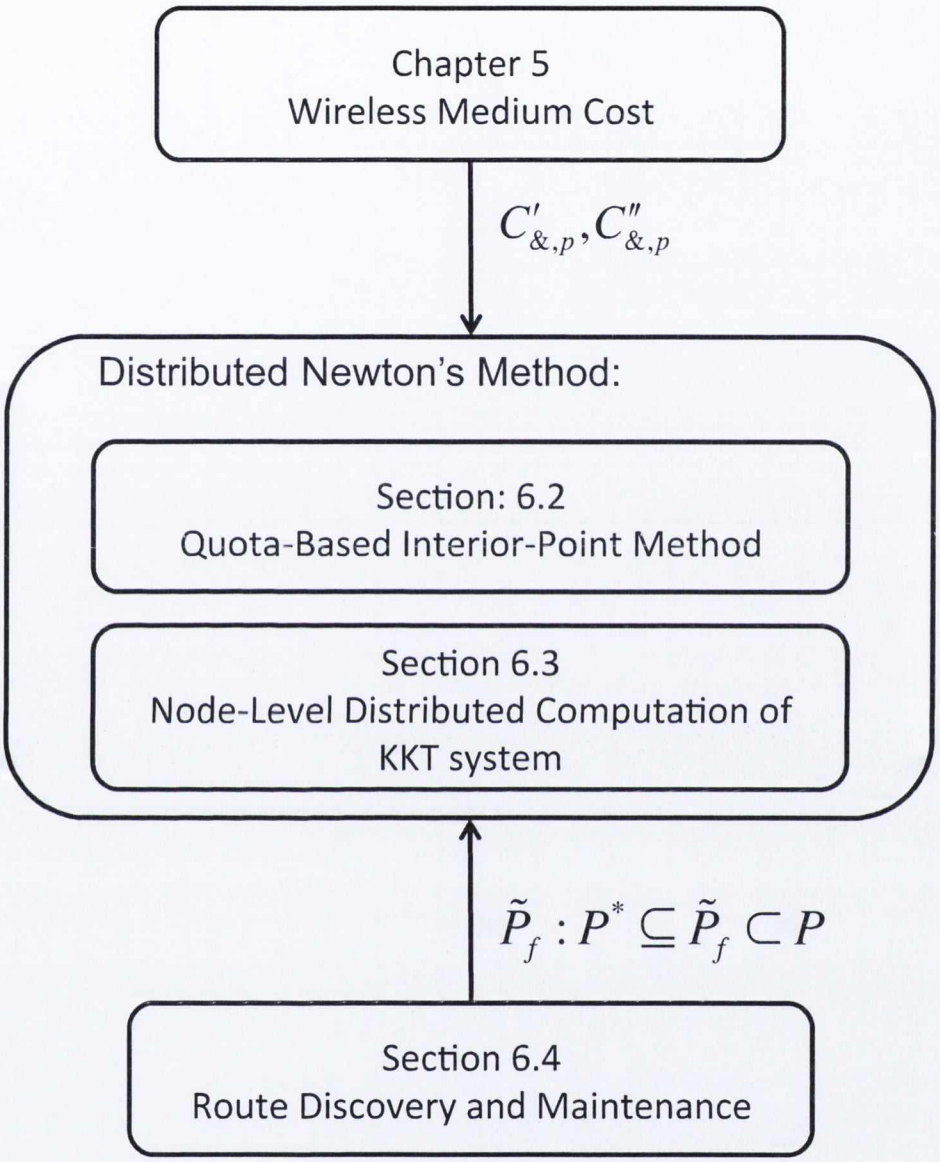


Fig. 5.1: Organization of Chapter 6

all time. On the other hand, constraints introduce couplings to different communication flows. In subsection 5.2.2, we propose an approach called the quota-based mechanism to decompose the couplings introduced by inequality constraints.

5.2.1 Addressing Constraints

The equality constraints can be tackled by equality-constrained Newton's method, namely solving the KKT system (3.19) at each Newton step. Computationally, equality constraints impose no difficulty to each step. This is known as the affine-invariant property of Newton's method. In the following, we focus on approaches that address the inequality constraints.

As we have reviewed in chapter 3, available tools for inequality constraints include the Frank-Wolfe method, the projection method, the interior-point method and decomposition methods. Among these tools, the Frank-Wolfe method is applied to the first-order gradient method; Decomposition methods can solve subproblems locally using the second-order Newton's method. However, the coupling constraints among different subproblems are addressed using the subgradient method. Both methods suffer from a slow convergence in addressing constraints. Furthermore, decomposition methods do not guarantee strict feasibility. Therefore, we focus on the projection method and the interior-point method.

The projection method guarantees strict feasibility and is simplistic to implement. However its crude "cut-off" on solutions may lead to fluctuations of convergence. Interior-point methods, including the basic logarithm barrier-based interior-point method and the primal-dual interior-point method, enjoy a fast convergence compared to the rest of the approaches. The problem for the primal-dual interior-point method is that it does not guarantee feasibility of the converging sequence of solutions. The logarithm barrier method is mostly satisfying except that it only applies to inequality on open sets.

The majority of existing routing approaches choose a single method to address all inequality constraints uniformly. In the contrary, we investigate the constraints formu-

lated in (1.2), namely the non-negative constraints and the capacity constraints, and address them separately.

5.2.1.1 Non-negative Constraints

Traffic flow along a path can be either zero if not utilized or positive if utilized, but it can not be negative, i.e. $\mathcal{F} \succeq 0$. This is equivalent to non-negative aggregated flows along each link, i.e. $\mathcal{F}^{(l)} \succeq 0$.

This constraint is not a strict inequality in that elements in $\mathcal{F}$ can be zero. By default, the interior-point method, shown in (3.22) does not suit this case: The penalty introduced by the logarithm barrier function, $-\mu log(\mathcal{F})$ is infinity at $f_p = 0$ for any path p. That means all paths have to have a positive traffic, i.e. being utilized. Nonetheless, this issue can be handled by an engineering modification to the constraints: For a small value $\epsilon > 0$, one can change the non-negative constraints $\mathcal{F} \succeq 0$ to $\mathcal{F} \succ -\epsilon$. As the coefficient μ of the barrier function in approaches to 0, the penalty $-\mu log(\mathcal{F})$ at $\mathcal{F} = 0$ diminishes. Therefore, paths can have zero traffic. However, it requires many outer iterations of the interior-point for the coefficient μ to approach 0. That means the abandonment of "bad" paths is slow.

On the other hand, the projection method works on strict inequalities. In addition, we argue that a projection on $\mathcal{F} \succeq 0$ is less likely to suffer from fluctuation. According to Corollary 2, we know that at network optimum, for each communication flow, all paths with higher marginal costs than the rest have zero traffic and vice versa. For the sake of argument, let us consider an example where a path p is transmitting flow $f_p = 2$. At some step, Newton's method determines a flow change $\Delta f_p = -5$. Because traffic over all paths sum to the traffic demand of a communication flow, projecting the flow change to $\Delta f_p = -2$ will result in less traffic on other paths of the same source. Since the marginal cost of a path decreases as its traffic decreases, roughly speaking, the projection leads to a even higher difference in marginal costs between path p and the rest of currently transmitting paths. Therefore, path f_p should stay un-utilized unless its marginal cost

reduces in the future. We choose the projection method on the non-negative constraints $\mathcal{F} \succeq 0$.

5.2.1.2 Capacity Constraints

Now we consider the capacity constraints, i.e. $g(\mathcal{F}) < \mathcal{C}$, which differs in wireless networks compared to wired networks. According to our analysis on wireless interference in section 4.2, we classify the capacity constraints into two categories: 1) the medium capacity constraints as a result of secondary interferences and 2) the queueing capacity constraints as a result of primary interferences.

Firstly, we have the medium capacity:

$$f_i^{(l)} + \sum_{f_j^{(l)} \in \mathcal{I}_{a,i}} f_j^{(l)} < \mathcal{C}_{a,i}; \quad \forall a \in \mathcal{N}, \forall i \in \mathcal{F}_a \tag{5.1}$$

where $\mathcal{C}_{a,i}$ is the capacity/maximum throughput at the medium of link i at node a. It states that the sum of traffic sharing the medium access is constrained by the medium capacity. The medium capacity constraint corresponds to the link capacity in wired networks.

Another type of capacity constraints is the queueing capacity constraints. Because all outgoing flows share the same queue at a node, the aggregated arrival rate of all flows can not overflow the queue. This can be formalized according to the queueing model (4.2): The utilization of the queue at node a should be strictly less than one:

$$\sum_{f_i^{(l)} \in \mathcal{F}_a} \ell_{a,i}^{(s)}(\mathcal{M}_{a,i}) f_i^{(l)} < 1; \quad \forall a \in \mathcal{N} \tag{5.2}$$

Each link flow $f_i^{(l)}$ can be transformed to a summation of path flows running through the link:

$$f_i^{(l)} = \sum_{\substack{p: p \in \mathcal{P} \\ i \in p}} f_p \tag{5.3}$$

(5.1), (5.2) and (5.3) together form the capacity constraints $g(\mathcal{F}) < 0$, where $g(\cdot) : \mathbb{R}^{|\mathcal{P}|} \rightarrow \mathbb{R}^{|\mathcal{N}|+|\mathcal{L}|}$. We can somehow number each of the $|\mathcal{N}| + |\mathcal{L}|$ constraints. Let $g_i(\mathcal{F})$ be the i^{th} constraint.

The detail of $g(\mathcal{F})$ depends on the choice of the medium access delay function $\ell_{a,i}^{(s)}(\cdot)$. But unless $\ell_{a,i}^{(s)}(\cdot)$ is either a constant function or the inverse of $f_i^{(l)}$, $g(\mathcal{F})$ is likely to be nonlinear, which may lead to complex computation. To simplify the problem, although the rest of our approach does not mandate such a simplification, we can replace the capacity constraints with a lower bound. Let $\hat{\ell}^{(s)}$ be the medium access delay of the underlying MAC approach at maximum throughput. Since the medium access delay is an increasing function, we know that $\hat{\ell}^{(s)}$ is the upper bound of $\ell_{a,i}^{(s)}(\cdot)$. Then, (5.2) becomes a linear inequality:

$$\sum_{f_i^{(l)} \in \mathcal{F}_a} \hat{\ell}^{(s)} f_i^{(l)} < 1; \quad \forall a \in \mathcal{N} \tag{5.4}$$

Combining (5.1), (5.3) and (5.4), the capacity constraints can be written in the linear form $g(\mathcal{F}) = \mathcal{C} - \mathcal{R}\mathcal{F} < 0$, although $\mathcal{R}\mathcal{F} \neq \mathcal{F}^{(l)}$. This constraint is indeed a lower bound on the actual capacity constraint. Routing solutions to problems under such a constraints are conservative in congestion avoidance.

In both the original and the linear approximated cases, the capacity constraint is not a strict inequality. We choose the interior-point method that introduces a logarithm barrier function $-\mu log(-g(\mathcal{F})) = -\mu \sum_{i=1}^{|\mathcal{N}|+|\mathcal{L}|} log(-g_i(\mathcal{F}))$ to the objective function in order to keep the solution sequence of Newton's method within the capacity constraints.

In summary, we choose a combination of the projection method and the interior-point method for constrained optimization. In next subsection, we discuss our design for distributed implementation of these methods.

5.2.2 Decoupling Constraints

In the previous subsection, we have discussed our choice on methods to tackle constraints. Our goal in this subsection is to decouple the constraints with respect to each commu-

nication flow. Firstly, let us examine the couplings of constraints. Then, we explain our rationale to decompose the couplings. Finally, we describe our proposed solution, called the *quota-based mechanism.*

The equality constraints are introduced from the traffic demand of each communication flow, as stated in our problem formulation1.2b: $\sum_{p \in \mathcal{P}_w} f_p = \mathcal{T}_w, \forall w \in \mathcal{W}$. Therefore, couplings exist between paths of the same communication flow rather than between different communication flows. More specifically, the equality-constrained Newton's method solves the KKT system (3.19) at each step. In every row i of the KKT matrix such that its corresponding path $p_i \in \mathcal{P}_w$ belonging to the same communication w, there exists one and only one dual variable ν_w, which can be solved locally at each source node. It is also easily known that the non-negative constraints $\mathcal{F} \succeq 0$ do not impose couplings between different paths or different communications.

Indeed, the only couplings of constraints among different communications are originated from the capacity constraints: Path flows that share the medium access (5.1) or share a queue (5.2) affect the capacity residual of each other. In fact, different communication are more strongly coupled in wireless communication compared to that in wired networks. 1) due to the shared queue at each node, the number of constraints are larger than that in wired communication - $|\mathcal{N}| + |\mathcal{L}|$ compared to $|\mathcal{L}|$. 2) Each constraint $g_i(\cdot)$, $\forall i = 1...|\mathcal{N}| + |\mathcal{L}|$, due to wireless interference, may involve more flows compared to link capacity constraints in wired networks. In the following we describe the rationale of a quota-based mechanism that decomposes the strongly coupled constraints.

As we have discussed in the previous subsection 5.2.1.2, we choose the interior-point method for capacity constraints, which introduces a barrier function on each of the $|\mathcal{N}| + |\mathcal{L}|$ constraints. The aggregated barrier function $-log(-g(\mathcal{F}))$ transforms the

capacity constraints to the overall network costs.

$$\begin{aligned} \text{minimize} \quad & C(\mathcal{F}) - \mu \sum_{i=1}^{|\mathcal{N}|+|\mathcal{L}|} log(-g_i(\mathcal{F})) \\ \text{subject to} \quad & \sum_{p \in \mathcal{P}_w} f_p = \mathcal{T}_w \quad w \in \mathcal{W} \\ & \mathcal{F} \succeq 0 \end{aligned} \tag{5.5}$$

Instead of decomposing the aggregated barrier function, which suffers from the complication of the barrier function, we can coordinate different flows sharing a queue or medium access by assigning a portion of the available capacity, namely a quota, to each flow. The assignment of quotas for each resource decouples different communication flows. It can be carried out at each intermediate node locally without evaluating the complicated barrier functions. Once the quotas of all queues and links are assigned for a path, only the minimum quota needs to be guaranteed. At the source node of each path, a single barrier function is applied locally to the least quota along the path.

We term such an approach the *quota-based mechanism*. Now, we give a formal description.

$$\begin{aligned} \underset{\mathcal{Q},\mathcal{F}}{\text{minimize}} \quad & C(\mathcal{F}) - \mu \sum_{p:f_p \in \mathcal{F}} log(\mathcal{Q}_p - f_p) \\ \text{subject to} \quad & \sum_{p \in \mathcal{P}_w} f_p = \mathcal{T}_w \quad w \in \mathcal{W} \\ & \mathcal{F} \succeq 0 \\ & g_i(\mathcal{Q}) < 0 \quad \forall i = 1...|\mathcal{N}| + |\mathcal{L}| \end{aligned} \tag{5.6}$$

where $\mathcal{Q} = \{\mathcal{Q}_p |\ p : f_p \in \mathcal{F}\}^T$[1] is the vector of quotas for all the path flows. It is worth noting that changing the last part of inequalities to equalities, i.e. $g_i(\mathcal{Q}) = 0, \quad \forall i = 1...|\mathcal{N}|+|\mathcal{L}|$, does not alter the solution to the problem. In the case where $g_i(\cdot)$ is a linear function, the equality constraints are preferred as Newton's method is affine invariant;

[1]Or equivalently, $\mathcal{Q} = \{\mathcal{Q}_p |\ p : \mathcal{P}\}^T$. We take the flow $\mathcal{F}$ point of view because F is our handle variable

If $g_i(\cdot)$ is nonlinear, the inequality constraints are preferred, as an interior-point method can be applied again. Modified from the interior-point method, both cases of the quota-based formulation can be shown to converges to the optimal solution to the original routing optimization problem.

Proposition 3. *As $\mu \to 0$, the solution to problem 5.6, denoted by $\tilde{\tilde{\mathcal{F}}}^*$ approaches to the solution to problem 5.5 denoted by $\tilde{\mathcal{F}}^*$.*

Therefore, the modified interior-point method (5.6) optimizes the original problem (1.2). Compared to the basic interior-point method (5.5), the couplings of constraints can be efficiently separated through a primal decomposition: The couplings between different f_p in the complicated $g(\mathcal{F})$ are reduced to $|\mathcal{P}|$ amount of couplings between $\mathcal{Q}_p$ and f_p. Through primal decomposition, each node computes locally its assignment of quotas to different flows. For each flow, only the minimum quota across different nodes is chosen. In this way, couplings between different $\mathcal{Q}_p$ are eliminated.

In the following, we give a mathematical description of the decomposition to the quota-based formulation (5.6).

Let $\kappa(a)$ be a subset of indices among $1...|\mathcal{N}| + |\mathcal{L}|$ that includes: 1) the index of the queueing capacity constraint for node a and 2) the indices of the medium capacity constraints for outgoing links of node a. Therefore, the set of quotas $\mathcal{Q}^a = \{\mathcal{Q}_p^a\}_{f_p \in \mathcal{F}_a \cup \mathcal{I}_a}$ [2] computed by node a for each f_p must meet constraints $g_i(\mathcal{Q}) < 0, \forall i \in \kappa(a)$. Indeed, any $\mathcal{Q}_{p'}^b \in \mathcal{Q}$ such that $b \neq a$ must have a zero coefficient in $g_i(\mathcal{Q})$. Without loss of generality, we can write the capacity constraint as $g_i(\mathcal{Q}^a) < 0$. Given the current traffic flows $\mathcal{F}^{(t)}$, the assignment of $\mathcal{Q}^a$ at node a can be executed locally as a subproblem, as

[2] With a slight abuse of notation, we let $f_p \in \mathcal{F}_a \cup \mathcal{I}_a$ denote flow f_p runs through node a or interferes with outgoing flows of node a. Namely, $f_p \in \mathcal{F}_a \cup \mathcal{I}_a$ if and only if there exists $f_i^{(l)} \in \mathcal{F}_a \cup \mathcal{I}_a$ such that $i \in \mathcal{L}_p$

shown here:

$$\begin{aligned} \underset{\mathcal{Q}^a}{\text{minimize}} \quad & -\sum_{f_p \in \mathcal{F}_a \cup \mathcal{I}_a} log(\mathcal{Q}_p^a - f_p^{(t)}) \\ \text{subject to} \quad & g_i(\mathcal{Q}^a) < 0 \quad \forall i \in \kappa(a) \end{aligned} \tag{5.7}$$

Solving this problem gives the $\mathcal{Q}^a$. In addition, let $\mathcal{Q}_p^a = \infty$ if f_p does not occur in any of node a's constraints, i.e. $f_p \notin \mathcal{F}_a \cup \mathcal{I}_a$. The quota of each path flow, $\mathcal{Q}_p$ can be aggregated from the quotas assigned to f_p by all nodes whose resources are used by f_p, namely, $\mathcal{Q}_p = \min\{\mathcal{Q}_p^a | \ \forall a \in \mathcal{N}\} = \min\{\mathcal{Q}_p^a | \ \forall a : f_p \in \mathcal{F}_a \cup \mathcal{I}_a\}$. To do so, intermediate nodes of a path can gather quotas assigned to the path flow from all its two-hop neighbours. Acknowledgement messages of the path flow propagate from the destination to the source node and keep a field of quota information. Along the propagation, this field is updated by the minimum quota assigned and gathered by all intermediate nodes. Cost information of the path can be forwarded in conjunction with the propagation of the quota information. When the aggregated quota $\mathcal{Q}_p$ and the first and second derivatives of medium costs are gathered for all paths, the traffic flows can be updated to $\mathcal{F}^{(t+1)}$. This is essentially one Newton step to the following problem:

$$\begin{aligned} \underset{\mathcal{F}}{\text{minimize}} \quad & C(\mathcal{F}) - \mu \sum_{p: f_p \in \mathcal{F}} log(\mathcal{Q}_p - f_p) \\ \text{subject to} \quad & \sum_{p \in \mathcal{P}_w} f_p = \mathcal{T}_w \quad w \in \mathcal{W} \\ & \mathcal{F} \succeq 0 \end{aligned} \tag{5.8}$$

Problem (5.7) and problem (5.8) are essentially the subproblem and the master problem of the quota-based formulation (5.6), decomposed by a primal decomposition.

Our quota-based mechanism are related to both decomposition techniques and to the interior-point method. In the next subsection, we will present a case study of our approach. Then we discuss the similarity and differences of our approach compared to existing distributed approaches, such as the distributed Newton's method and existing decomposition methods.

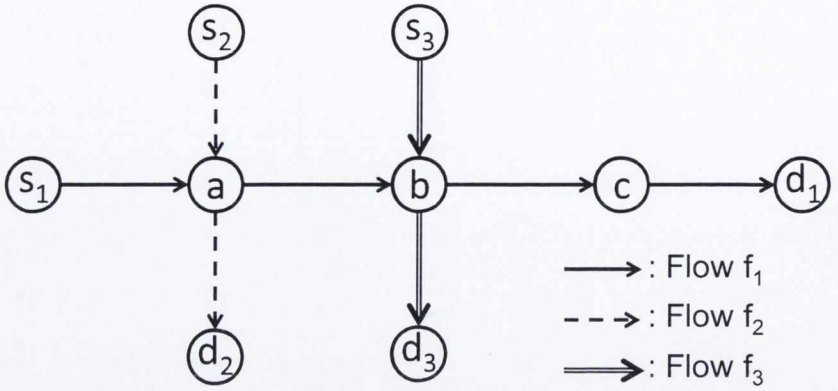


Fig. 5.2: An example scenario to illustrate the quota-based mechanism: Flow f_i is identified by a $(s_i - d_i)$ pair, $i = 1, 2, 3$. Throughputs of a, b, c are capacitated.

5.2.3 An Example

In order to illustrate the operations of the quota-based approach, let us now consider a hypothetical network, shown in Fig. 5.2. In this network, we use the link model in wired networks to simplifiy the scenario: Intermediate nodes a, b and c each have a throughput capacity, $\mathcal{C}_a, \mathcal{C}_b$ and $\mathcal{C}_c$. Therefore the capacity constraint $g_i(\mathcal{F}) < 0$ is of a linear form: For $i = a, b$ and c, $g_i(\mathcal{F}) = \sum f_p - \mathcal{C}_i < 0$. There are three single-path communication flows. Since our focus in this section is to decouple constraints using the quota-based approach, we assume a separable objective function for $C(f_1, f_2, f_3) = -\sum_p log(f_p)$. The problem to be optimized is given as:

$$\begin{aligned} \underset{f_p}{\text{minimize}} \quad & -\sum_{p=1,2,3} log(f_p) \\ \text{subject to} \quad & f_1 + f_2 < \mathcal{C}_a; \quad f_1 + f_3 < \mathcal{C}_b \\ & f_1 < \mathcal{C}_c \end{aligned} \tag{5.9}$$

Note that the non-negative constraints are omitted in this formulation as the objective function $log(f_p)$ already dictates the positive nature of flows. Transforming it to quota-

based formulation, we have:

$$\begin{aligned} \underset{f_p, \mathcal{Q}_p}{\text{minimize}} \quad & -\sum_{p=1,2,3} log(f_p) - \mu \sum_{p=1,2,3} log(\mathcal{Q}_p - f_p) \\ \text{subject to} \quad & \mathcal{Q}_1 + \mathcal{Q}_2 = \mathcal{C}_a; \quad \mathcal{Q}_1 + \mathcal{Q}_3 = \mathcal{C}_b \\ & \mathcal{Q}_1 = \mathcal{C}_c; \end{aligned} \tag{5.10}$$

Since the inequality constraints are linear, we choose $g_i(\mathcal{Q}) = 0$ in the quota-based formulation. Each $\mathcal{Q}_p$ is coupled with a flow variable f_p in the objective function. A primal decomposition can be applied: The subproblem is to compute the best $\mathcal{Q}$ given a traffic flow step $\mathcal{F}^{(t)}$; With the solutions of all subproblems, the master problem can compute the next Newton step $\mathcal{F}^{(t+1)}$. For example, the subproblem at node a is

$$\begin{aligned} \underset{\mathcal{Q}_1^a, \mathcal{Q}_2^a}{\text{minimize}} \quad & -log(\mathcal{Q}_1^a - f_1^{(t)}) - log(\mathcal{Q}_2^a - f_2^{(t)}) \\ \text{subject to} \quad & \mathcal{Q}_1^a + \mathcal{Q}_2^a = \mathcal{C}_a \end{aligned} \tag{5.11}$$

Using equality constrained Newton's method, this problem can solved locally at node a, returning the value of $\mathcal{Q}_1^a$ and $\mathcal{Q}_2^a$. Similarly node b and c computes their quota assignment. The quota of a path is aggregated from the assignment at all nodes.

$$\mathcal{Q}_1 = \min\{\mathcal{Q}_1^a, \mathcal{Q}_1^b, \mathcal{Q}_1^c\}; \quad \mathcal{Q}_2 = \mathcal{Q}_2^a; \quad \mathcal{Q}_3 = \mathcal{Q}_3^b$$

When $\mathcal{Q}_1$, $\mathcal{Q}_2$ and $\mathcal{Q}_3$ arrives at source nodes of the three communications, the master problem can be computed distributively at each source node. For $p = 1, 2, 3$, the master problem is as follows:

$$\underset{f_p}{\text{minimize}} - log(f_p) - \mu log(\mathcal{Q}_p - f_p) \tag{5.12}$$

Let $y(f_p)$ be the objective function, i.e. $y(f_p) = -log(f_p) - \mu log(\mathcal{Q}_p - f_p)$. Using Newton's method, the update of f_p is given as:

$$f_p^{(t+1)} = f_p^{(t)} - \alpha \frac{y'(f_p^{(t)})}{y''(f_p^{(t)})}$$

Then a new round (t+1) of transmission and computation starts. Once the computation of (5.12) terminates, μ is updated as $\mu = \beta\mu$ where $\beta > 1$. As $\mu \to 0$, the solution $f_p \to f_p^*$.

One interesting property of this particular formulation (5.9) is that the objective cost function $-log(f_p)$ can be evaluated locally at the source node of f_p. Therefore, an alternative primal decomposition of problem (5.10) is to make the problem (5.12) the subproblem and solve locally at each source node. Then the problem (5.11) becomes the master problem. $\mathcal{Q}_p$ is updated one Newton step with each round of communication.

5.2.4 Summary and Discussion

So far in this section, we have presented our approach to address constraints in routing optimization problems. Namely, we answer the first open question identified in section 3.8.2:

What is a suitable distributed approach to address the couplings introduced by equality and inequality constraints?

We choose the equality-constrained Newton's method for the traffic demand constraint of each communication and the projection method for non-negative constraints. The capacity constraints $g(\mathcal{F}) < 0$ have a different form in wireless networks compared to wired networks, due to wireless interferences. We classify the capacity constraints into the medium capacity constraints and the queueing capacity constraints. To address the capacity constraints, we choose the barrier function-based interior-point method over the projection method and the dual decomposition technique for its fast convergence.

The main focus of this section is to decouple the capacity constraints with respect to each communication flows. To do so, we introduce the quota-based mechanism that extends the centralized interior-point method. In brief, at each intermediate where multiple flows are jointly constrained by a certain capacity, a quota is assigned to each of the flow. Quotas of at all intermediates travel back to the source node of each path.

In this way, the couplings are eliminated since each path flow is only constrained by its quotas. An interior-point method can be used to find the next round of traffic. The assignment of quotas at each intermediate is also an optimization problem, whose objective function can be evaluated locally. The algorithm starts with any feasible solution. Given a traffic flow at each round, the assignment of quotas can be solved locally at each node by equality-constrained Newton's method.

The quota-based approach can be vaguely classified as an indirect decomposition method [Palomar and Chiang, 2006], which introduces auxiliary variables to assist decomposition. However, compared to existing distributed approaches, such as decomposition methods and distributed Newton's method, it has three desirable features:

Firstly, the quota-based mechanism integrates the interior-point method with decomposition techniques. The benefit is twofold: On one hand, it is capable of addressing coupled nonlinear constraints. Namely, $g_i(\mathcal{F}) < 0$ can take the form as (5.2). In the contrary, the dual decomposition method, although it is a natural approach to decouple the constraints, relies on the separability of constraints. For example, a common form of constraints addressed by dual decomposition is $\mathcal{R}\mathcal{F} < 0$. On the other hand, the logarithm barrier function enjoys a quick adjustment to inequality constraints. If a solution is approaching the boundaries, the penalties increase superlinearly therefore pushing the solution within feasible domain. To the best of our knowledge, only one approach exists in the domain of network optimization that combines the interior-point method with decomposition techniques [Mosk-Aoyama et al., 2010]. However, their approach lacks the following features of our approach.

Secondly, both the subproblems and the master problem in our formulation can be addressed using the second-order Newton's method. Recall that in general decomposition methods, the coordination between different subproblems, i.e. the master problem, is executed with the gradient method - if with luck the master problem is differentiable - or with the subgradient method - if the master problem is not differentiable. The first-order coordination of subproblems results in a slow convergence. Distributed Newton's method

uses iterative methods, including the matrix splitting technique and the GaBP method, to distributively compute the linear system at each Newton's step. Although the outer Newton's iteration retains the quadratic convergences, a large number of iterations may be inserted at each step, which slows down the overall convergence. In comparison, our approach is more efficient in addressing coupled constraints.

Thirdly, the quota-based approach aggregates quotas assigned by intermediate nodes of a path into the quota of the path. The aggregation is essentially a reduction to the set of constraints for a path flow. As a result, only the most binding capacity constraint of a path flow is returned to the source node and used in the logarithm barrier function. As we have seen previously, due to wireless interferences, different communication flows are heavily coupled in ad hoc networks: 1) Two flows may interfere with each other at multiple hops with different types of interferences; 2) A flow may be interfered by a large amount of other flows. Both the standard decomposition approaches and the distributed Newton's methods suffer from a slow convergence rate in heavily coupled problems. On the other hand, the quota-based approach addresses the optimization problem with reduced couplings. Therefore, it is more suitable to ad hoc networks compared to existing approaches.

The quota-based mechanism exhibits features that are more suitable for strongly coupled ad hoc networks compared to existing distributed algorithms. It can be anticipated these features result in a fast convergence of the quota-based approach. We will confirm this in chapter 6.

In summary, we have described our design choice to address constraints and proposed the quota-based mechanism to decouple capacity constraints with respect to each communication flows. In section 3.8.2, we have summarized three open questions for the design of optimal routing algorithms in ad hoc networks. The design in this section essentially answers the first question. In the next section, we try to answer the rest two open questions.

5.3 Node-level Distributed Manner

In the previous section, we propose the quota-based mechanism to decouple constraints in the optimal routing problem with respect to each communication flow. We note that in general, the quota-based mechanism alone can not decouple an optimization problem, unless the objective function of the optimization problem is separable. For example, if the objective is the utility function, $-\sum_p log(f_p)$, the quota-based mechanism is sufficient to facilitate a flow-level distributed algorithm. Another example is using link flows $\{f_i^{(l)}\}$ instead of path flows as handle variables . Then, the objective function in the optimal routing problem becomes $\sum_{i\in\mathcal{L}} C_i(\pi_{a,i})$. This function is not mathematically separable with respect to each link flow $f_i^{(l)}$, in the sense that $\pi_{a,i}$ consists of not only $f_i^{(l)}$ but also neighbouring link flows $f_j^{(l)} \in \mathcal{F}_a \cup \mathcal{I}_{a,i}$. Nonetheless, coupled link flows are physically close to each other. Therefore, the couplings introduced by the cost function can be eliminated by exchanges of information locally. In this case, the quota-based mechanism is sufficient to facilitate a node-level distributed routing algorithm.

However, both cases do not suit optimal routing problems in wireless ad hoc networks. As we have discussed previously in section 3.2.3, the utility function does not model the interference in between different flows; As we have discussed previously in section 3.8.2, using link flows as handle variables increases the size of the problem. It slows down the convergence of distributed algorithms and introduces more communication overheads.

Therefore, we focus the minimization of the non-separable cost function $C(\mathcal{F})$ using path flows as handle variables, $\mathcal{F} = \{f_p |\ p \in \mathcal{P}\}$. Firstly, in subsection 5.3.1, we investigate the design choice to decouple the cost function in Newton's method with respect to each path flows. Together with the quota-based mechanism, it leads to a flow-level distributed routing algorithm. Then, in subsection 5.3.2, we propose the *tree-based mechanism* that translates the flow-level distributed algorithm into a node-level distributed algorithm.

5.3.1 Diagonal Approximation of the Hessian

In this subsection, we propose to use the diagonal approximation of the Hessian matrix in Newton method. In this way, the couplings between different communication flows can be eliminated at each Newton step. In the following, we firstly describe the motivation of this approach. Then we describe an integration of the diagonal approximation with the quota-based mechanism. Finally, we discuss the advantages and disadvantages of this approach.

5.3.1.1 Motivation

The couplings introduced by a non-separable objective function $C(\mathcal{F})$ are embodied by the computation of the Newton step. For example, in the unconstrained case, we have

$$\nabla^2 C(\mathcal{F})\Delta\mathcal{F} = -\nabla C(\mathcal{F})$$

Let $[\nabla^2 C(\mathcal{F})^{-1}]_i$ denote the i^{th} row of the inverse of the Hessian matrix $\nabla^2 C(\mathcal{F})$. The i^{th} path flow is given by $\Delta f_i = [\nabla^2 C(\mathcal{F})^{-1}]_i \nabla C(\mathcal{F})$. In general, the inverse operation requires a full knowledge of the Hessian matrix and a centralized computation. In order to compute the inverse of the Hessian matrix distributively, a common approach is to use iterative methods to solve the coupled linear equation system. However, these approaches introduces a number of iterations at each Newton step, which introduces communication overheads and does not suit wireless ad hoc networks. We have identified this challenge as the second open question for the design of a distributed optimal routing algorithm, as stated in section 3.8.2.

How can we address the coupling introduced by the objective function, namely the distributed inverse of the Hessian coefficient matrix, without using iterative methods at each Newton step?

We answer this question by using the diagonal approximation of the Hessian matrix in the computation of each Newton step. Let $diag(\nabla^2 C(\mathcal{F}))$ represent the matrix whose diagonal elements are the same as the Hessian $\nabla^2 C(\mathcal{F})$ and all off-diagonal elements are

set to zero. The unconstrained Newton step using diagonal approximation is given as :

$$diag(\nabla^2 C(\mathcal{F}))\Delta\mathcal{F} = -\nabla C(\mathcal{F})$$

In this case, each component can be computed by $\Delta f_i = \frac{\partial^2 C(\mathcal{F})}{(\partial f_i)^2}^{-1} \frac{\partial C(\mathcal{F})}{\partial f_i}$. Now that the couplings introduced by the objective function are eliminated, we can integrate the diagonal approximation with the quota-based approach for a flow-level distributed optimal routing algorithm.

5.3.1.2 Flow-level Distributed Algorithm

Combining the results we have so far, including the wireless medium cost, the quota-based mechanism and the diagonal approximation of the Hessian matrix, we can describe a flow-level distributed routing algorithm.

The algorithm starts with an initial traffic $\mathcal{F}^{(0)}$. Each source node iteratively adjusts its traffic distribution over available paths. After the round t transmission, each intermediate node a gathers information of neighbouring traffic flows. Then, it calculates the first and the second derivative medium costs, namely $C'_{\&,a}(f_p)$ and $C''_{\&,a}(f_p)$, of each path flow f_p running through it. It solves problem (5.7) using Newton's method, which gives the assignment of quotas for each path flow. A reply message from the destination node to the source node records the derivatives of medium cost and quotas at each intermediate node. When the reply message arrives from path p, the source extracts $C'_{\&,p}(f_p)$, $C''_{\&,p}(f_p)$ and $\mathcal{Q}_p$.

For simplicity of notation, we can number the path flows of one communication w as $f_{1,w} \cdots f_{n,w}$. With the information extracted from the reply message at round t, the source

of communication w computes its Newton step from the following equation systems:

$$\begin{cases} \left(C''_{\&,1,w}(f^{(t)}_{1,w}) + \frac{\mu}{\left(\mathcal{Q}_{1,w}-f^{(t)}_{1,w}\right)^2}\right)\Delta f^{(t)}_1 + \nu_w = -\left(C'_{\&,1,w}(f^{(t)}_{1,w}) + \frac{\mu}{\mathcal{Q}_{1,w}-f^{(t)}_{1,w}}\right) \\ \qquad\qquad \vdots \\ \left(C''_{\&,n,w}(f^{(t)}_{n,w}) + \frac{\mu}{\left(\mathcal{Q}_{n,w}-f^{(t)}_{n,w}\right)^2}\right)\Delta f^{(t)}_n + \nu_w = -\left(C'_{\&,n,w}(f^{(t)}_{n,w}) + \frac{\mu}{\mathcal{Q}_{n,w}-f^{(t)}_{n,w}}\right) \\ \sum_{i=1}^{n} \Delta f^{(t)}_{i,w} = \Delta \mathcal{T}_w/\alpha \end{cases} \tag{5.13}$$

where $\Delta \mathcal{T}_w = 0$ if the traffic demand of communication w is consistent otherwise represents the change of traffic demand compared to the previous round. The next round of traffic distribution is given as

$$f^{(t+1)}_{i,w} = f^{(t)}_{i,w} + \alpha \Delta f^{(t)}_{i,w}, \quad i = 1...n \tag{5.14}$$

where α is the step size and can be either a constant positive $0 < \alpha \leq 1$ or determined by certain line search algorithms.

The path flows computed by (5.13) and (5.14) may be negative. As we have discussed, we choose the projection method to enforce the non-negative constraints. Projection methods used in existing network optimization approaches are linear projections. In this case, a linear projection sets the negative path flows as zero and equally distributes the negative residual over positive paths, which is rather simplistic. A more sophisticated approach is to distribute the residual over the remaining paths based on their path costs.

More specially, let r be the sum of path flows at round t that becomes negative after updated from (5.13) and (5.14). Namely,

$$r = \sum_{i \in NegSet} f^{(t)}_{i,w}; \quad \text{where} \quad NegSet = \{i |\ f^{(t+1)}_{i,w} < 0\} \tag{5.15}$$

Firstly, the source node sets path flows whose index belongs to *NegSet* un-utilized, i.e. assigning zero traffic over previously negative flows. Secondly, it removes equations of un-utilized paths in (5.13) and replace the last equation by $\sum_{i=1}^{n} \Delta f^{(t)}_{i,w} = (\Delta \mathcal{T}_w - r)/\alpha$. Then, system (5.13) and (5.14) are computed once again for a new traffic distribution.

Finally, it is possible that the new traffic distribution also contains negative flows. In this case, the same procedure can be executed iteratively until no negative flows occurs. It is easily known that this iteration always terminates, i.e. it is not possible that all traffic flows are assigned negative traffic. It should be noted that due to the diagonal approximation, the computational effort needed to solve system (5.13) is trivial.

This traffic distribution algorithm is essentially the master problem (5.8) at each source node. We use Newton decrement as the criterion for the stopping criterion of the problem. That is,

$$\lambda^2(\mathcal{F}_w) = -\sum_{i=1}^{n} \left(C'_{\&,i,w}(f_{i,w}^{(t)}) + \frac{\mu}{\mathcal{Q}_{i,w} - f_{i,w}^{(t)}} \right) \Delta f_{i,w}^{(t)} \tag{5.16}$$

The inner Newton iteration to problem (5.8) terminates if λ^2 is smaller than a certain threshold. Then the interior-point method continues with a smaller μ. We give a formal description in algorithm 3.

Algorithm 3 Load Distribution Algorithm at Source Nodes

Starting with an initial value $\mathcal{F}_w^{(0)}$, $\nu^{(0)} > 0$, $\epsilon^{interior} > 0$, $\epsilon^{decrement} > 0$ and $\gamma < 1$

repeat:

1. Send out traffic $\mathcal{F}_w^{(t)} = \{f_{1,w}^{(t)}, ..., f_{n,w}^{(t)}\}$
2. Gather cost and quota information $C'_{\&,i,w}(f_{i,w}^{(t)})$, $C''_{\&,i,w}(f_{i,w}^{(t)})$ and $\mathcal{Q}_{i,w}$ for each path i

3 **repeat:**

3.1 Solve system (5.13) and update the solution with (5.14)

3.2 If there exists $f_{i,w}^{(t+1)} < 0$, then computes $NegSet$ and r according to (5.15)

3.3 Set $f_{i,w}^{(t+1)} = 0$ and remove corresponding equation in (5.13) for every $i \in NegSet$

3.4 Set $\Delta\mathcal{T}_w = \Delta\mathcal{T}_w - r/\alpha$

until all path flow $f_{i,w}^{(t+1)} \geq 0$

4. Compute the Newton decrement $\lambda^2(\mathcal{F}_w)$ according to (5.16)
5. Update the $\mu = \gamma\mu$ if $\lambda^2(\mathcal{F}_w) < \epsilon^{decrement}$

until the stopping criterion $n\mu < \epsilon$ is met

5.3.1.3 Discussion

Indeed, the usage of diagonal Hessian matrix is a simplistic technique to avoid the expensive computation of Newton step. Certain routing algorithms [Bertsekas et al., 1984] have also applied diagonal Hessian to facilitate distributed computation of Newton step. Compared to Newton's method using the full Hessian matrix, the diagonal approximated Newton's method can not guarantee quadratic convergence in general. However, we argue that the diagonal approximation of the Hessian matrix is a suitable approach for optimal routing in wireless ad hoc networks.

Firstly, it avoids the necessity of iterative methods to compute each Newton step. At each Newton iteration, the computation can be carried out locally with one round

of data-reply message exchange between the source node and the destination node. Secondly, it reduces the amount of information needed at each iteration. Since off-diagonal elements of the Hessian matrix are omitted, each intermediate node gathers and propagates less information compared to the full Hessian. The reduction overhead can be significant in a network with non-sparse communication flows. Finally, although the diagonal approximated Hessian matrix can not a quadratic convergence, it exhibits a good convergence rate in practise. In fact, we can view the diagonal Hessian as a preconditioner to the gradient of network costs. Therefore, the approximate Newton's method using diagonal Hessian guarantees a linear convergence rate and avoids the "zig-zagging" problem experienced by the gradient method. Without "zig-zagging", the geometrically reducing costs of linear converging algorithm is usually satisfying.

5.3.2 Node-Level Distributed Optimization based on Optimal Substructure

So far, we have given the design of a flow-level distributed routing algorithm where the source node of each communication makes load distribution decisions over available paths. Indeed, the flow-level distributed routing algorithm can be seen as the multipath version of source-routing, which is undesirable for ad hoc networks. In order to achieve a node-level distributed algorithm, existing approaches have focused on the pure mathematical side in the sense that they formulate the routing optimization problem with link rates as handle variables. In this way, the same optimization methods can be applied to node-level distributed algorithms as to the flow-level algorithms.

In addition to what described in the mathematical formulations, the routing problem possesses an interesting property, namely the *optimal substructure*. With this property, we can design a node-level distributed optimal algorithm. Firstly, we reiterate the definition of the optimal substructure property [Cormen et al., 2009]:

A problem exhibits optimal substructure if an optimal solution to the problem contains within it optimal solutions to subproblems.

The concept of the optimal substructure originated from the domain of dynamic programming and greedy algorithms. It should be noted that the term "optimal" here differs from that in the domain of convex optimization. In the case of multipath routing optimization using iterative methods, we argue that the computation of each step $\mathcal{F}^{(t)}$ exhibits the optimal substructure in the sense that the solution to the system (5.13) contains sub-solutions at each intermediate node.

Let us consider the example network shown in Fig. 5.3(a). At each round, the task is to split the traffic demand over the three paths based on their cost information - $C'_{\&,p}(f_p), C''_{\&,p}(f_p), \mathcal{Q}_p, p = 1, 2, 3$. Physically, data traffic of flow f_1 and f_2 are separated from f_3 at source node S at first. Then f_1 and f_2 are split at node A. In a more complex network, data traffic gradually separated as they approach the destination node. Therefore, the traffic splitting problem consists of a sequence of subproblems, each of which splits partially the traffic demand. Each subproblem is solved based on the solution of its previous subproblems, i.e. it exhibits the optimal substructure.

Now we give the mathematical realization of the optimal substructure property. We simplify the notation in system (5.13) as follows:

$$\tilde{C}'^{(t)}_p = \left(C'_{\&,p,w}(f^{(t)}_{p,w}) + \frac{\mu}{\mathcal{Q}_{p,w} - f^{(t)}_{p,w}}\right)$$

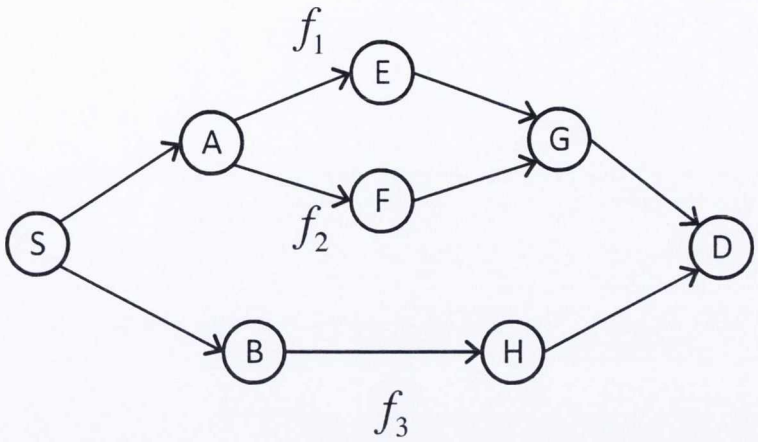
and

$$\tilde{C}''^{(t)}_p = \left(C''_{\&,p,w}(f^{(t)}_{p,w}) + \frac{\mu}{\left(\mathcal{Q}_{p,w} - f^{(t)}_{p,w}\right)^2}\right)$$

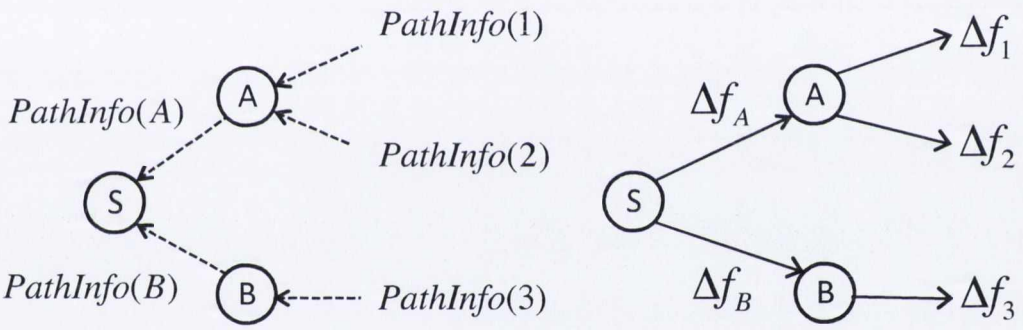
Then, in this example, the traffic distribution problem at step t is given as:

$$\begin{cases} \tilde{C}''^{(t)}_1 \Delta f^{(t)}_1 + \nu_w = -\tilde{C}'^{(t)}_1 \\ \tilde{C}''^{(t)}_2 \Delta f^{(t)}_2 + \nu_w = -\tilde{C}'^{(t)}_2 \\ \tilde{C}''^{(t)}_3 \Delta f^{(t)}_3 + \nu_w = -\tilde{C}'^{(t)}_3 \\ \Delta f^{(t)}_{1,w} + \Delta f^{(t)}_{3,w} + \Delta f^{(t)}_{3,w} = \Delta \mathcal{T}_w/\alpha \end{cases} \tag{5.17}$$

Let Δf_A and Δf_B be the solution of the subproblem at source node S. The subproblem of problem (5.17) at node S is



(a) An Example Network: Three available paths, f_1, f_2 and f_3 for communication between node S and D



(b) Substructure of the Routing Problem: Dashed lines represent reverse paths for reply messages to source nodes. $PathInfo(p) = \{C'_{\&,p}, C''_{\&,p}, \mathcal{Q}_p\}$

Fig. 5.3: A Demonstration of Optimal Substructure for Multipath Routing: At each round of an iterative optimal routing method, the computation of routing decisions can be seen as a traffic splitting problem. The optimal splitting at each round consists of an optimal splitting between f_A and f_B at node S and an optimal splitting between f_1 and f_2 at node A. The optimality of solutions to each subproblem depends on the proper merging of cost information.

$$\begin{cases} \tilde{C}_A^{''(t)} \Delta f_A + \nu_w = -\tilde{C}_A^{'(t)} \\ \tilde{C}_B^{''(t)} \Delta f_B + \nu_w = -\tilde{C}_B^{'(t)} \\ \Delta f_A + \Delta f_B = \Delta \mathcal{T}_w / \alpha \end{cases} \tag{5.18}$$

For the optimal substructure property to hold, the system (5.18) should give a solution $\Delta f_A = \Delta f_1 + \Delta f_2$ and $\Delta f_B = \Delta f_3$. According to (5.17), we have

$$\begin{aligned} \Delta f_A = & \Delta f_1 + \Delta f_2 = \frac{-\tilde{C}_1^{'(t)} - \nu_w}{\tilde{C}_1^{''(t)}} + \frac{-\tilde{C}_2^{'(t)} - \nu_w}{\tilde{C}_2^{''(t)}} \\ = & \frac{-\tilde{C}_1^{'(t)} \tilde{C}_2^{''(t)} - \tilde{C}_1^{''(t)} \tilde{C}_2^{'(t)} - \left(\tilde{C}_1^{''(t)} + \tilde{C}_2^{''(t)} \right) \nu_w}{\tilde{C}_1^{''(t)} \tilde{C}_2^{''(t)}} \end{aligned}$$

, which simply transforms to

$$\frac{\tilde{C}_1^{''(t)} \tilde{C}_2^{''(t)}}{\tilde{C}_1^{''(t)} + \tilde{C}_2^{''(t)}} \Delta f_A + \nu_w = -\frac{\tilde{C}_1^{'(t)} \tilde{C}_2^{''(t)} + \tilde{C}_1^{''(t)} \tilde{C}_2^{'(t)}}{\tilde{C}_1^{''(t)} + \tilde{C}_2^{''(t)}} \tag{5.19}$$

That means, if we let

$$\tilde{C}_A^{'(t)} = \frac{\tilde{C}_1^{'(t)} \tilde{C}_2^{''(t)} + \tilde{C}_1^{''(t)} \tilde{C}_2^{'(t)}}{\tilde{C}_1^{''(t)} + \tilde{C}_2^{''(t)}}, \quad \tilde{C}_A^{''(t)} = \frac{\tilde{C}_1^{''(t)} \tilde{C}_2^{''(t)}}{\tilde{C}_1^{''(t)} + \tilde{C}_2^{''(t)}} \tag{5.20}$$

, then (5.19) becomes the first equation in system (5.18). Also let $\tilde{C}_B^{'(t)} = \tilde{C}_3^{'(t)}$ and $\tilde{C}_B^{''(t)} = \tilde{C}_3^{''(t)}$. Solving system (5.18) gives $\Delta f_A = \Delta f_1 + \Delta f_2$ and $\Delta f_B = \Delta f_3$.

Since node S needs only $\tilde{C}_A^{'(t)}$ and $\tilde{C}_A^{''(t)}$ but not necessarily cost information of path f_1 and f_2, equation (5.20) can be seen as the specification of the merging of path f_1 and f_2 should merge at node A.

Receiving aggregated traffic from S, i.e. based on the solution of the first subproblem, node A can compute the split of f_1 and f_2 as follows:

$$\begin{cases} \tilde{C}_1^{''(t)} \Delta f_1 + \nu_w = -\tilde{C}_1^{'(t)} \\ \tilde{C}_2^{''(t)} \Delta f_2 + \nu_w = -\tilde{C}_2^{'(t)} \\ \Delta f_1 + \Delta f_1 = \Delta f_A \end{cases} \tag{5.21}$$

The traffic splitting problem terminates with f_1, f_2 and f_3 computed. This completes the mathematical realization of the optimal substructure property. The procedure is demonstrated in Fig. 5.3(b).

It should be noted that there is one difference in the information exchange between node-level distributed manner and flow level distributed manner. If the algorithm uses aggregated quota of a path, the merge of paths requires the path quota instead of the node quota. Therefore, instead of gathering quota information on in reply messages from destinations to sources. Intermediate nodes maintain their quota assignments during the continuous transmission of one round of data. The quota-assignment for next round of transmission can be piggybacked to destination nodes with data messages. $\tilde{C}_p^{'(t)}$ and $\tilde{C}_p^{''(t)}$ can be computed starting from source nodes and updated along reverse paths.

Generalizing the derivation of this example, we give the description of a node-level distributed routing algorithm.

$$\begin{cases} C'_{\&p} = \dfrac{\sum\limits_i C'_{\&p_i} \prod\limits_{j \neq i} C''_{\&p_j}}{\sum\limits_i \prod\limits_{j \neq i} C''_{\&p_j}} \\ C''_{\&p} = \dfrac{\prod\limits_i C''_{\&p_i}}{\sum\limits_i \prod\limits_{j \neq i} C''_{\&p_j}} \end{cases} \tag{5.22}$$

Receiving the cost information of node A and node B, the source node S can now calculate the changes of flow sending to the next-hop nodes as in (5.13)

Intermediate node A, on receiving the change of traffic flow, calculate the change of sub-flows to its next-hop neighbours as in (5.23)

$$\begin{cases} C''_{\&p_1} \Delta f_1 + w = -C'_{\&p_1} \\ C''_{\&p_2} \Delta f_2 + w = -C'_{\&p_2} \\ \cdots\cdots \\ C''_{\&p_n} \Delta f_m + w = -C'_{\&p_m} \\ \sum\limits_{p_i} \Delta f_i = \Delta f_A \end{cases} \tag{5.23}$$

The algorithm at intermediate nodes is given as follows:

Algorithm 4 Algorithm at Intermediate Nodes

On receiving data messages:

1. Compute the traffic distribution to the next hop neighbours according to system (5.23). If no cost information is available, distribute traffic to all next hop neighbours evenly.
2. During the continuous transmission of current round traffic, update its quota assignment by solving problem (5.7)

On receiving reply message of one communication:

1. Merge the cost information according to (5.22)
2. Update local cost information to the merged cost information:
$$C''_{\&p} = C''_{\&p} + C''_{\&a}; \quad C'_{\&p} = C'_{\&p} + C'_{\&a}$$
3. Forward reply messages back to last hop nodes.

5.4 Route Discovery

In this section, we investigate the route discovery procedure for optimal routing algorithms. Firstly, in subsection 5.4.1, we revisit the overview of the network layer communication task for optimal routing. Our node-level distributed quota-based interior-point algorithm offers partly the functionalities of the overall task. We identify the primary remaining issues, namely the discovery and establishment multiple loop-free paths, to facilitate our optimal routing algorithm. In section 5.4.3 and 5.4.2, we describe our approach to address these issues.

5.4.1 Overview

The overall communication task of data transmission between source nodes and destination nodes using multiple paths can be divided into four components: the control

message exchange, path filtering, path selection, and data distribution. Existing optimal routing algorithms assume the availability of all potential paths and focus on the computation of traffic assignment. Essentially they provide the path selection and data distribution functionalities. Therefore, our objective is to design an approach that discovers and establishes multiple loop-free paths for optimal routing algorithms. Raised from the literature review in section 2.5.4 and 2.5.5, this approach should have the following properties:

1. The ability to eliminate redundant path information, by path filtering and merging.
2. Discovery of optimal path set if potential paths are filtered.
3. Providing exact cost information, which is challenging if multiple paths are merged.

Indeed, our design objective have already been partly addressed by results described so far. Our node-level distributed quota-based interior-point algorithm uses cost metric $C'_{\&,p}(f_p)$, $C''_{\&,p}(f_p)$ and $\mathcal{Q}_p$ for each path p. Therefore, a natural control message exchange mechanism is the distance-vector routing. Sources node floods route request (RREQ) messages. Multiple - if not all - RREQ messages are replied once reach destination nodes. Route reply (RREP) messages travel the reverse path of RREQ messages to source nodes. Routes between sources and destinations are established. The on-demand manner suits the scenario of dense ad hoc networks because the global network topologies are not necessarily maintained at all nodes as link-state routing. In our node-level distributed algorithm, The merging of cost information, as specified in (5.22), is exact in the sense that it does not affect the resulting traffic assignment over each path compared to unmerged costs. Finally, same as other optimal routing algorithms, path selection and data distribution are determined by iterative convex optimization methods.

One outstanding issue yet to be addressed is the path filtering during route discovery. A proper path filtering criterion eliminates redundant paths as much as possible. Therefore, the communication and computation overheads are reduced compared to naive

route discovery approaches. We will search for such a criterion in subsection 5.4.3.

Another issue is that although we have specified the exact merging of costs, it may conflict with the establishment of loop-free paths. As we have discussed in chapter 2, in order to prevent loops during data forwarding, the cost to a destination node has to be strictly decreasing as the data approaching the destination node. Previously in section 2.5.5, we argued that this may not always be the case if costs are merged. In subsection 5.4.2, we will investigate this issue.

5.4.2 Loop-freedom of Multipath Routing

In this subsection, we discuss our approach to guarantee loop-freedom in multipath communication. The problem can be divided into two aspects: 1) loop-freedom during route discovery stage, i.e. in a static setting; 2) loop-freedom during route maintenance, i.e. with the presence of topology changes.

5.4.2.1 Loop-Freedom of RREQ Floodings

Firstly, we consider the discovery of loop-free paths in a static setting. This is not an issue for single-path distance-vector routing algorithms. Only one path is to be discovered per communication. If an intermediate node receives multiple copies of a RREQ message identified by the source and destination addresses, it forwards only the first copy and discards all consecutive copies. Disjoint multipath routing is a similar case. Each intermediate node - or link - transmits only one copy of a RREQ message and discards the rest. Feasibility condition based multipath routing maintains a single shortest path between each node and the destination node. Therefore its route discovery procedure is essentially a proactive single path routing. However, the discovery of loop-free paths is not obvious for our case. Since we do not assume the path disjointness, multiple paths may intersect at one intermediate node. An intermediate node may have to forward multiple copies of RREQ messages for one communication. The question is, how can an intermediate node determine whether a received copy of RREQ messages is

a result of loops during flooding or a loop-free intersecting path?

A straightforward solution to this problem is to record the list of nodes travelled by a copy of RREQ message. On receiving the RREQ message, an intermediate node can check whether its address is already in the list. A positive result indicates the RREQ message has travelled a loop. Otherwise, the intermediate node rebroadcasts the RREQ message. However, the list of intermediate nodes travelled by a message can take a large amount of space in the header of the message, which introduces communication overheads.

In order to alleviate the overheads, we can replace the list of addresses with a small *bloom filter* in the header of RREQ messages. A bloom filter is a randomized data structure to determine the membership of an element in a given set. It consists of k hash functions that maps an element to a m-bit binary string. In our case, the IP addresses of network nodes are the elements whose membership to be determined. The usage of bloom filters involves two operations: add and query. All the m bits are initially set to 0 in a bloom filter. When a RREQ message arrives at an intermediate node for the first time, the intermediate performs the"add" operation: The IP address of the node is fed to the k hash functions, each of which returns a m-bit string. Then a bit-wise OR operation is taken over all the k strings and the original string in the bloom filter. Finally, the node updates the string in the bloom filter with the result of the OR operation. In order to check whether the arrival of a RREQ message is the first time, an intermediate node performs the "query" operation to the bloom filter: The node feeds its IP address to all the k hash functions and performs OR over the k binary strings, which returns an array of positions whose value is 1. If the value of these positions in the string of bloom filter are all 1, then the result of the query is deemed as positive. It indicates that the RREQ message has travelled through the node previously. The node discards the RREQ message. Otherwise, the node performs an "add" operation to insert its IP address to the bloom filter of the RREQ message and rebroadcast the RREQ message to all its neighbours.

Indeed, many previous approaches [Broder and Mitzenmacher, 2004,Tian and Cheng, 2012] have used bloom filters to eliminate loops of messages. False negative results are not possible in bloom filters, which guarantees loop-freedom. On the other hand, false positive results are possible in bloom filters. This is because all the corresponding bits of one element in the bloom filter can be set by a group of other elements. This leads to a false positive query to that element. A false positive result eliminates a loop-free path falsely. However, the probability of false positive is controllable. In the following, we briefly state the analysis.

Let n be the expected maximum length of a path. For a bloom filter with m-bit string and k perfectly randomized hash functions [3], the probability of false positive, denoted by p, is given as follows:

$$p = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k$$

Since p can be seen as a function of k given the value of m and n, there exists a k^* that minimizes the false positive probability.

$$k^* = \frac{m}{n} log(2)$$

With the optimal k determined, we can write:

$$m = -\frac{nlog(p)}{\left(log(2)\right)^2}$$

Therefore,given the design objective of the probability of false positive and the expected maximum path length, we can determine the number of bits in the bloom filter. For example, if we allow a 0.1% of false positive and 20 hops of longest path, the bloom filter will take up 288 bits in the header. That is on average 14.38 bits overhead introduced by each hop, compared to the 32 bits of addresses for each RREQ message.

[3]Perfect randomization here implies that each bit is set independently with each other. In practise, for example, MD5 is commonly used

5.4.2.2 Loop-Freedom in a Dynamic Setting

The usage of bloom filters eliminates loops in the flooding of RREQ messages. If the topology of a network is ideally static, paths established by RREQ messages will stay loop-free. However, ad hoc networks may experience topology changes due to node movement or joining and leaving a network. Although in our thesis we do not consider highly dynamic environments, such as mobile ad hoc networks, we wish to cope with infrequent topology changes, i.e. quasi-static networks.

With the presence of topology changes, established paths may break or change part of their courses. As we have reviewed in chapter 2, existing routing approaches establish an order of intermediate nodes, which decreases strictly towards a destination node. In this way, an intermediate node can determine its next-hop nodes based on the order value of its neighbours, which are not necessarily the next-hops established by RREQ messages. For example, in shortest path routing, hop counts towards destination nodes are used as the order value. In feasibility condition-based minimum delay routing [Vutukury and Garcia-Luna-Aceves, 1999], each node reports the minimum delay cost towards a destination to its upstream nodes. In this way, a strictly decreasing order of delay costs towards a destination is established. However, this results in an inexact merging of cost metrics of multiple paths.

The merging of cost metrics in our algorithm, as specified in (5.22), is exact. However, it is easily seen that a merged first derivative cost is essentially a weighted average of all its next-hop first derivative costs. Unless all next-hop first derivative costs are equal, there is at least one next-hop cost larger than the merged first derivative cost. A merged second derivative cost is always smaller than any of the next-hop second derivative costs. In short, the exact merging of cost metrics in our algorithm does not retain the strictly decreasing order among intermediate nodes towards destination nodes. It leads to loops in data forwarding.

In order to address the dilemma between exact merging of cost metrics and strictly

decreasing order, we differentiate the cost metric used by traffic distribution and the distance towards a destination. In addition to the derivative costs $\{C'_{\&,p}, C''_{\&,p}\}$ and quota information $\mathcal{Q}_p$ of a path, the RREP message also carries a *hop count* field that increases at each hop. The merging of hop counts from multiple paths chooses the smallest hop count value of each next-hop nodes. Therefore, a strictly decreasing order is established. Based on hop count value of each neighbours, a node determines its next-hop nodes, i.e. potential paths. Based on the cost metrics $\{C'_{\&,p}, C''_{\&,p}, \mathcal{Q}_p\}$, a node determines the data distribution over the selected nodes, using (5.23).

Finally, information of a path may become stale due to topology changes. Possible methods to identify outdated information include sequence number and feasibility condition. For its simplicity, we choose the approach of sequence number. A sequence number is associated with each communication. If a node detects link failure, it increases the current sequence number it holds by one and starts a local repair, i.e. route request by the node. The local repair procedure terminates when a node with a higher sequence number responses to its requests.

5.4.3 A Path Filtering Criterion for Optimal Routing

In the previous subsection, we have addressed the loop-freedom problem in both route discovery and data forwarding. Indeed, eliminating loops is a basic path filtering criterion: Paths with loops should always be avoided. On the other hand, not all paths without loops are necessary to discover and maintain. In this subsection, we investigate the path filtering criterion for optimal routing algorithms.

In a network with a high connectivity, such as a dense ad hoc network, the number of potential paths between two nodes can be astronomical. For example, even in a small scale 5*5 grid topology as shown in Fig. 5.4, there are 8512 potential paths between node 1 and node 25. On one hand, a dominating majority of the available paths will not be selected by an optimal routing algorithm. On the other hand, it is practically infeasible for an algorithm to discover all the potential paths. Therefore, a path filtering

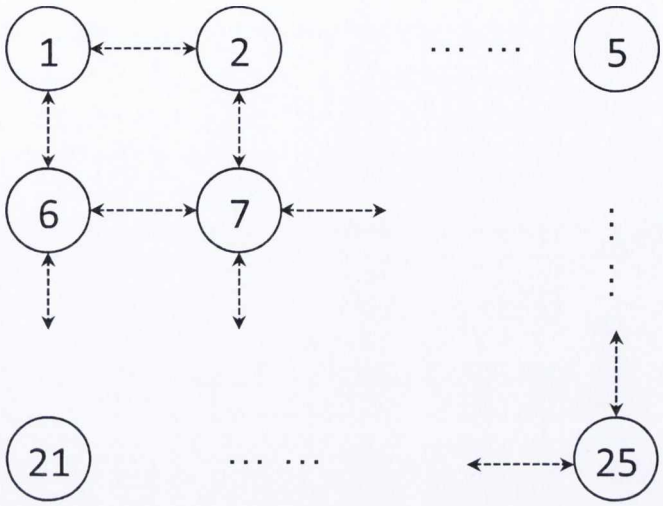


Fig. 5.4: A Demonstration of the Number of Potential Paths: In this 5x5 mesh network, there are 8512 paths without loops between node 1 and node 25.

criterion that eliminates redundant paths in the route discovery stage is necessary.

We have anticipated that a proper path filtering criterion should be based on the cost metric used by the optimal routing algorithm. This assertion now becomes obvious since we have established the optimal Wardrop equilibrium. According to the optimal Wardrop equilibrium, under an optimal routing decision, all the paths used by a communication have the same marginal cost which is no less than those un-utilized paths. That means, if the marginal cost of a path is too large compared to other paths, then it will likely be discarded by optimal routing algorithm. We can therefore set a time-to-live (TTL) field in the flooded RREQ messages. Assuming the size of a RREQ message is ϵ, at each hop the RREQ message travels, a marginal medium cost $C'_{\&,a}(\epsilon)$ is subtracted from the TTL field. Once the TTL value of a RREQ message dries out, the message is discarded as it indicates that the path travelled is too long in the derivative length. This manner draws a connection between the on-demand route discovery in classical routing algorithms, such as AODV, and optimization approaches.

The question is, what is the threshold of marginal costs in identifying redundant

paths, i.e. How can a source node determines a proper TTL value? We answer this question by giving the following proposition.

Proposition 4. *For each communication w, the solution it computes at each round t as $\Delta\mathcal{F}_w^{(t)}$ and $\nu_w^{(t)}$, from the diagonal approximated quota-based interior-point method (5.13). Let quota is $\mathcal{Q}^0$ denote the initial quota nodes assigned to a communication. Define an appended KKT system with a new path $\tilde{p}$, whose first and second derivative lengths are $C'_{\&,\tilde{p}}(f_{\tilde{p}})$ and $C''_{\&,\tilde{p}}(f_{\tilde{p}})$, as follows*

$$\begin{cases} \Big(C''_{\&,1,w}(f_{1,w}^{(t)}) + \frac{\mu}{\left(\mathcal{Q}_{1,w}-f_{1,w}^{(t)}\right)^2}\Big)\Delta f_1^{(t)} + \nu_w^+ = -\Big(C'_{\&,1,w}(f_{1,w}^{(t)}) + \frac{\mu}{\mathcal{Q}_{1,w}-f_{1,w}^{(t)}}\Big) \\ \qquad\qquad \vdots \\ \Big(C''_{\&,n,w}(f_{n,w}^{(t)}) + \frac{\mu}{\left(\mathcal{Q}_{n,w}-f_{n,w}^{(t)}\right)^2}\Big)\Delta f_n^{(t)} + \nu_w^+ = -\Big(C'_{\&,n,w}(f_{n,w}^{(t)}) + \frac{\mu}{\mathcal{Q}_{n,w}-f_{n,w}^{(t)}}\Big) \\ \Big(C''_{\&,\tilde{p}}(\epsilon) + \frac{\mu}{\left(\mathcal{Q}^0-\epsilon\right)^2}\Big)\Delta f_{\tilde{p}}^{(t)} + \nu_w^+ = -\Big(C'_{\&,\tilde{p}}(\epsilon) + \frac{\mu}{\mathcal{Q}^0-\epsilon}\Big) \\ \sum_{i=1}^n \Delta f_{i,w}^{(t)} + \Delta f_{\tilde{p}}^{(t)} = \Delta\mathcal{T}_w/\alpha \end{cases} \tag{5.24}$$

Then, the necessary condition for $\Delta f_{\tilde{p}}^{(t)} > 0$ computed from the appended KKT system (5.24) is

$$C'_{\&\tilde{p}}(\epsilon) < -\nu_w^{(t)} - \frac{\mu}{\mathcal{Q}^0-\epsilon} \tag{5.25}$$

Proof. See appendix D □

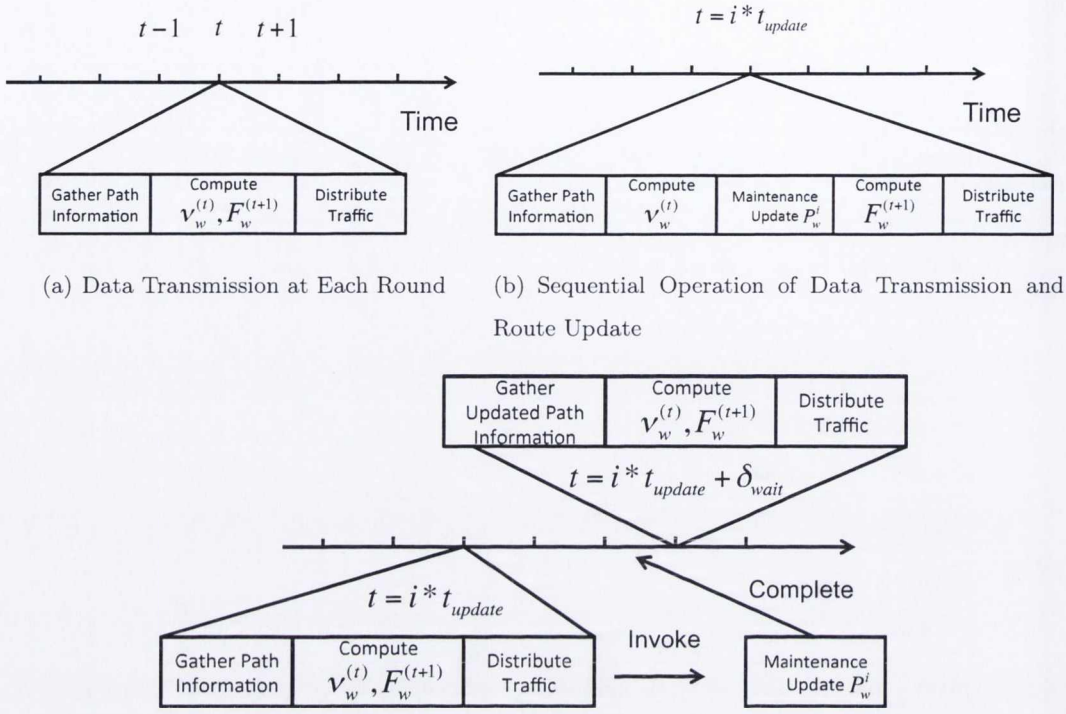
This proposition states that, if there is a path $\tilde{p}$ that has not been taken into consideration at current round of computation, for it to be utilized by the communication w under the cost information of round t, condition (5.25) has to be met. Therefore, the value $-\nu_w^{(t)} - \frac{\mu}{\mathcal{Q}^0-\epsilon}$ computed at the source node of communication w can be used as a threshold at round t. All paths whose derivative cost is larger than the threshold would be certainly discarded by the diagonal approximated quota-based interior-point method. Furthermore, although (5.25) is not a sufficient condition, it is indeed a strong filtering

criterion - any path that meets condition (5.25) has a smaller marginal cost than at least one path in existing path set. It is likely to filter out the majority of redundant paths.

Based on this proposition, we can design an iterative route discovery approach. It starts with a "trial and error" manner to find the initial path set $\mathcal{P}_w^0$. At step $t = 0$, where $\nu_w^{(0)}$ is not yet computed, the source node can assign a predefined value to the TTL field of its RREQ messages and flood the messages. If it does not hear any RREP messages, then it increases the TTL value and floods RREQ messages again. This procedure may execute multiple times until either it receives RREP messages or it determines that the destination is unreachable. Each RREP message received by the source node establishes a path - possibly merged - towards the destination node. The initial path set $\mathcal{P}_w^0$ for communication w is established. The quota-based interior-point method computes the data distribution $\mathcal{F}_w^{(t)}$ and $\nu_w^{(t)}$ over $\mathcal{P}_w^0$, $t = 0, 1, ...$

During the convergence, the source node can periodically invoke a path maintenance procedure. Assuming that the update period is t_{update}, the path set of communication w is updated by $\mathcal{P}_w^i$ at each time slot $t = t_{update} * i$, where $i = 1, 2, ...$ More specifically, after the computation of $\mathcal{F}_w^{(t+1)}$ and $\nu_w^{(t)}$, instead of transmitting data traffic, the source node floods a RREQ message with the TTL field set as $-\nu_w^{(t)} - \frac{\mu}{\mathcal{Q}^0 - \epsilon}$. All RREP messages that reach the source node establish the updated path set $\mathcal{P}_w^i$. Finally, the source node computes the data distribution $\mathcal{F}_w^{(t)}$ over the updated path set and starts transmission. In this way, the path set $\mathcal{P}_w^i$ will converge to the optimal path set $\mathcal{P}_w^*$ as $i \to \infty$. In practise, we observed rather quick convergence, i.e. much less rounds required compared to the convergence of data distribution.

An apparent issue of this route update procedure is that it blocks the data transmission at round t. This is because proposition 4 gives the necessary condition of adopting a new path at round t, given the result from computation of round t. The operation of data transmission without the procedure of route update is shown in Fig. 5.5(a). In comparison, the operation of data transmission is interrupted by the procedure of route update, as shown in Fig. 5.5(b). That is, the operations of data transmission and route



(a) Data Transmission at Each Round

(b) Sequential Operation of Data Transmission and Route Update

(c) Parallel Operation of Data Transmission and Route Update

Fig. 5.5: Time Axis of Data Transmission and Route Update: Differences between sequential update and parallel update

update are sequential. Ideally, we prefer a parallel manner of these operations. That is, the data transmission of round t does not wait for the update of path set. But rather, the paths discovered from the results of round t is used for the computation of round $t + \delta_{wait}$, where δ_{wait} is the number of round needed for the route discovery to complete, which usually is 1. This desirable operation is shown in Fig. 5.5(c).

The problem for this approach is that proposition 4 guarantees the optimality of paths discovered only in the sequential manner but not in the parallel manner. In the following, we demonstrate that a route update procedure parallel to the data transmission can also guarantee the optimality of paths discovered.

Firstly, we give a trivial observation on a threshold value that is independent to the coefficient μ of interior-point method.

Corollary 3. *Another necessary condition, which is slight weaker, is as follows:*

$$C'_{\&\tilde{p}}(\epsilon) < -\nu_w^{(t)}$$

Now, we give the long-term path filtering criterion which assists the parallel route update.

Corollary 4. *A necessary condition for a path $\tilde{p}$ to be in the optimal path set, i.e. $\tilde{p} \in \mathcal{P}^*$ is that there exists a $\tilde{t}$ and infinitely many $t > \tilde{t}$, such that*

$$C'_{\&\tilde{p}}(\epsilon) < -\nu_w^{(t)}$$

Corollary 4 states that if a path belongs to the optimal path set $\mathcal{P}_w^*$, it will pass the filtering threshold of infinitely many rounds. The union of all paths that pass the filtering threshold $\nu_w^{(t)}$ of any round t constitutes a path set set $\tilde{\mathcal{P}}_f$ such that $\mathcal{P}^* \subseteq \tilde{\mathcal{P}}_f$. Therefore, the approach of updating path set with $\mathcal{P}_w^i$ at round $t > i * t_{update}$ will iteratively converge to optimal path set. The optimality of the route update that is parallel to data transmission is justified.

In summary, we have proposed an iterative route discovery and update approach in this subsection. Instead of gathering information of all potential paths, which is often infeasible, our approach gradually builds up the set of optimal paths along the convergence of the optimal data distribution algorithm. We have established thresholds for path filtering. Proposition 4 and its corollaries demonstrate that optimal paths will not be eliminated from these thresholds. This is the desirable case of path filtering criterion that we have raised in chapter 2, for example as shown in Fig. 2.5(a).

5.5 Summary

In this chapter, we have presented the main design of this thesis. We propose a quota-based mechanism that addresses capacity-constraints with the interior-point method. We introduce auxiliary variables called quotas at each node. It represents the maximum traffic allowed of a path flow at a node. Each node optimizes the quota-assignment locally using second-order Newton's method. Quotas are forwarded to sources of communications, where a second-order interior-point step is computed. The operation of quota to each flow resembles Resource Reservation Protocol - Traffic Engineering (RSVP-TE) [Awduche et al., 2001] although the mathematical assignment of quotas is not address in RSVP-TE.

We use the diagonal approximation of the Hessian matrix in Newton step method, which separates the couplings in the objective function. In order to reach a node-level distributed manner, we utilize the optimal substructure property of routes between sources and destinations. Instead of breaking the network optimization problem into subproblems at each node as in existing distributed approaches, the flow-level distributed routing optimization can be carried out at each intermediate nodes recursively from destinations to sources.

We have also described the design of route discovery for our optimal routing algorithm. We presented a necessary condition for potential paths to be used by a communication, which can be used as a TTL value to filter out long or congested paths. We have applied a bloom filter to avoid loops in RREQ flooding. We propose to separate the order of nodes in a path from the cost metric of the paths. More specially, we use hop counts to determine the order of intermediate nodes in a path, and use derivative costs and quotas as metrics to compute traffic distribution.

Part III

Verifications of Proposed Solutions

Chapter 6

Evaluation and Discussion

In chapter 4 and 5, we have described our design to an optimal routing approach for wireless ad hoc networks, namely the quota-based interior-point routing algorithm using wireless medium cost (QBIP). In this chapter, we will present both analytical and statistical evaluation on the performance of QBIP.

In section 6.1, we give an overview of the evaluation strategy.

6.1 Overview

In the following, we conclude the purposes of our evaluation, that is, the focal points to be verified of the proposed approaches.

Firstly, although we have formulated the network optimum as $min.C(\mathcal{F})$, it is yet to be demonstrated whether and to what extend an optimal solution reduces costs in wireless networks compared to existing selfish approaches. Or to put it in a different perspective, how bad is selfish routing in wireless networks?

In section 6.2, we extend the analytical "price of anarchy" results in wired networks [Roughgarden and Tardos, 2002] to wireless networks. This extension identifies the worst case performance for an ideal selfish routing compared to optimal solutions – or equivalently the best case of cost reduction for optimal routing. We also perform

statistical simulations to illustrate the average cost reduction of network optimum compared to selfish routing decisions. We investigate the affects of different network factors to the average cost reduction. Results in section 6.2 justify the necessity and applications of wireless optimal routing approaches.

Secondly, we evaluate the performance of the proposed quota-based interior-point (QBIP) method. As we has discussed in section 3.1, two aspects of interests of an optimal routing algorithm are its convergence rate and the distributed manner. As summarized in algorithm 3 and 4, it is easily seen that QBIP is a node-level distributed algorithm. Therefore, the remaining question is how fast the node-level distributed QBIP converges. In section 6.3, we compare QBIP against some of the existing optimal approaches, both centralized and distributed.

The statistical evaluations in section 6.2 and section 6.3 are conducted in different types of simulations:

1. For the comparison between optimal solutions and selfish solutions, we constructed a conceptual network following the protocol model [Jain et al., 2003]. We implement two types of topologies: random and mesh. Routes are not presented in the start of simulations. Route discovery procedure, discussed in section 5.4, is used and provide potential paths for both optimal (QBIP) and selfish approaches.

2. In order to evaluate the convergence rate of QBIP against existing optimization approaches, we skip the route discovery stage and focus on solving the mathematically formulated routing problems (1.2). We have implemented several optimization approaches, including an iterative Newton's method via PCG, the dual decomposition, etc. and studied their performances in MATLAB [MATLAB, 2012].

Finally, we summarize the pros and cons of QBIP in comparison with existing optimization methods and identify the future work in section 6.4

6.2 How Good Is Wireless Optimal Routing?

In this section, we evaluate the performance gain for optimal routing solutions compared to ideal selfish routing solutions. An ideal selfish routing solution refers to the exact compliance with the selfish Wardrop equilibrium. More specifically, a routing solution is ideally selfish if for each communication w, given its traffic demand, all the paths it utilized have the same delay which is less than or equal to those un-utilized but available paths. We use the term "ideal" for such solutions for two reasons: 1) Given the traffic demand, each unit of data is routed 2) Without considering algorithms used, routing solutions at Wardrop equilibrium have been demonstrated to outperform other selfish routing solutions [Raghunathan and Kumar, 2009], such as single-path minimum delay routing or shortest path routing. The goal in this section is to demonstrate the necessity of optimal routing wireless networks.

We start with an extension to analytical results in wired networks.

6.2.1 An Analytical Result

In this subsection, we investigate analytically the worst case performance of selfish solutions in comparison to optimal solutions in wireless networks. That is, we establish the upper bound for the value $\frac{C(\mathcal{F}_{\text{selfish}})}{C(\mathcal{F}^*)}$, for any network topology and traffic demands of all communications. For a given class of latency function $\ell(\cdot)$, this upper bound is usually referred to as the"price of anarchy". In his seminal articles [Roughgarden and Tardos, 2002] and [Roughgarden, 2002], Roughgarden establishes some fundamental results for the anarchy value in wired networks. In the following, we summarize some of the important related results:

1. The price of anarchy is independent of the network topology. To compute the price of anarchy for a certain class of latency function, it suffices to compute the anarchy value in a two-node, two-link network where one of the links has a constant latency

function[1].

2. With a non-decreasing continuous latency function, the cost incurred by an ideal selfish solution for given traffic demand is at most the cost incurred by an optimal solution for twice of the traffic demand. i.e. the anarchy value is $max.\frac{C(\mathcal{F}_2^*)}{C(\mathcal{F}^*)}$, where $|\mathcal{F}_2^*| = 2|\mathcal{F}^*|$

3. The price of anarchy value can be given by $\sup_{r>0:\ell(r)>0}[\lambda\mu + (1-\lambda)]^{-1}$, where $\lambda \in (0,1) \cap \{\lambda | \ell'(\lambda r) = \ell(r)\}$, $\mu = \frac{\ell(\lambda r)}{\ell(r)} \in [0,1]$. The anarchy value MAY be infinitely large for certain cost functions. For instance, for polynomial delay function $\sum_{i=0}^{i=p} a_i x^i$, the price of anarchy is given by $\frac{(p+1)\sqrt[p]{p+1}}{(p+1)\sqrt[p]{p+1}-p}$.

These results are based on the assumption of link model in wired networks. In particular, the result number 3 does not follow the same form in wireless networks. We extend the analysis to wireless networks.

Consider a conceptual network as shown in Fig. 6.1, where there are two communications. The first communication has a traffic demand of r to route from node A to node D. Suppose latency over path A-B-C-D is constant, i.e. $\ell_1(\cdot) = c$; and latency over path A-E-F-D is given by $\ell_2(\cdot)$. The second communication has a traffic demand f_3 to send from G to E, which interferes with the transmission from E to F.

Given r and f_3, there exists increasing $\ell_2(\cdot)$ such that $\ell_2(r + f_3) = c$. Since $\ell_2(\cdot)$ is increasing, $\ell_2(f_3) < \ell_2(r + f_3) = c = \ell_1(0)$. The adaptive selfish approach will route all traffic to link 2, resulting an overall cost: $c \cdot r + \ell_3(r + f_3)f_3$.

Because

$$C'_{\&2}(f) = \ell'_2(f + f_3) + \ell_2(f + f_3) + \ell'_3(f + f_3)f_3 > \ell_2(f + f_3)$$

we have

$$C'_{\&2}(r) > \ell_2(r + f_3) = c = C'_{\&1}(r) \tag{6.1}$$

If we choose $\ell_3(f_3)f_3 > c$, then

[1]The rationale of using the constant latency function can be vaguely understood by the fact that a constant function belong to any given class of latency function, for example by setting the coefficient approximates to zero. See Roughgarden's original paper for detailed analysis and discussions.

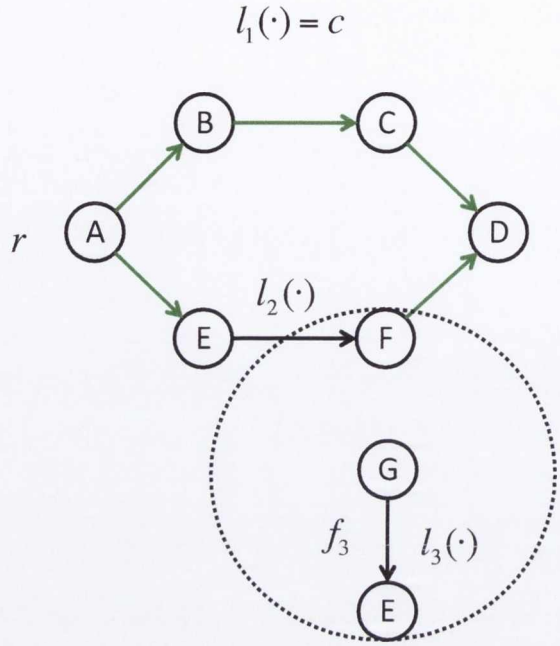


Fig. 6.1: A Conceptual Scenario to Prove the Price of Anarchy: Suppose the capacity at green links are arbitrarily large so that delay experienced is constant.

$$C'_{\&2}(0) = \ell_2(f_3) + \ell'_3(f_3)f_3 > c = C'_{\&1}(r) \tag{6.2}$$

Combining (6.1) and (6.2), we know that an optimal solution routes all traffic through link 1, resulting an overall cost: $c \cdot r + \ell_3(f_3)f_3$. The anarchy value α is given as:

$$\alpha = \frac{\ell_3(r + f_3)f_3 + c \cdot r}{\ell_3(f_3)f_3 + c \cdot r}$$

Choosing carefully c, r and f_3 that meet $\ell'_3(f_3)f_3 > c$. For example, as the constant value $r \to \infty$, f_3 is fixed, choose $c = r^{-1}$. It can be easily shown that the upper bound of α is infinity for any function $f_3(\cdot)$ that grows to infinity as its input grows, for example, polynomial functions and queueing delay functions. In summary, we have

$$\sup_{r,c,f_3} \alpha = +\infty$$

The price of anarchy describes the cost of selfish solutions using optimal solutions as benchmark. An equivalent metric that evaluates the optimal solutions is the cost reduc-

tion of optimal solutions compared to selfish solutions, which is defined as $\frac{C(\mathcal{F}_{\text{selfish}})-C(\mathcal{F}^*)}{C(\mathcal{F}_{\text{selfish}})} \times 100\%$. As the anarchy value approaches infinity, the cost reduction approaches 100%.

6.2.2 Implementation and Experiment Settings

Now that we have established analytically the best case cost reduction of optimal routing solutions compared to selfish solutions, some natural questions arise: 1) What is the average cost reduction of optimal routing solutions? 2) What are the factors that dictate the average cost reduction? For example, although the network topology plays no role in the upper bounds of the cost reduction, does it affect the *average* cost reduction over a number of runs? We wish to investigate these questions via statistical evaluations.

We implement the experiments following the protocol model: Nodes are placed within a conceptual disc. Each node has a transmission range of R, which for simplicity reason is also the interference range. Two nodes can communicate with each other directly if and only if their distance $d < R$. All communications within the interference range of each other contribute their delay experienced.

The placement of nodes may follow either the mesh topology or the random topology. In the mesh topology, the transmission/interference range of a node cover either 4 neighbours (orthogonal) or 8 neighbours (orthogonal and diagonal), as shown in 6.2. In the random topology, N_{nd} nodes are randomly placed in a disc with fixed area S. We can set up a network with an expected node density ND, which is defined as the average number of neighbours of each hop. The expected density can realised by setting the transmission range $R = \sqrt{\frac{S \cdot \text{ND}}{\pi \cdot N_{nd}}}$. Denote the number of communications in a network as N_{comm}, each of which is defined by a source and destination pair and has a traffic demand of D_w.

In each run of simulations, a network, mesh or random, is generated with given parameters. At first, a route discovery process is used to discover all the potential paths for each communication. Then, an optimal routing algorithm and an ideal selfish routing algorithm are used in the same scenario for comparison of their solutions. We

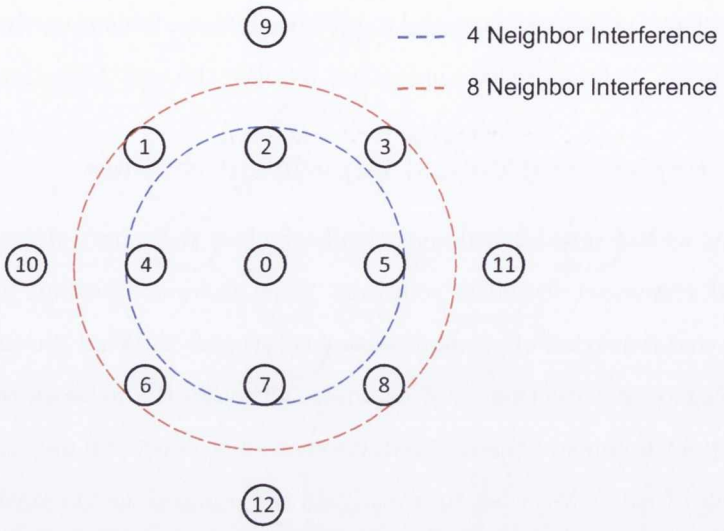


Fig. 6.2: Two Types of Interference Ranges Used in Mesh Topology

do not evaluate their convergence rates. Each result is aggregated from a minimum of 20 samples.

We use the proposed QBIP algorithm to compute the optimal solutions. As we will see in section 6.3, QBIP may not converge to the exact network optimum in certain scenarios due to the usage of diagonal approximated Hessian matrix. Nonetheless, QBIP is the only feasible routing algorithm in wireless ad hoc networks in the sense that it is node-level distributed and the error is comparably small.

The ideal selfish routing solutions are computed with a heuristic load-balancing algorithm: At each iteration, each source node reduces traffic load on paths with comparatively large delay and increases traffic load on paths with comparatively small delay. Iterations stop only if for each communication, paths utilized have the same delay.

6.2.3 Factors of Cost Reduction - Mesh Topology

Our first set of experiments are conducted in a 5×5 mesh network, with two communications whose sources and destinations are randomly selected. We choose this small-scale

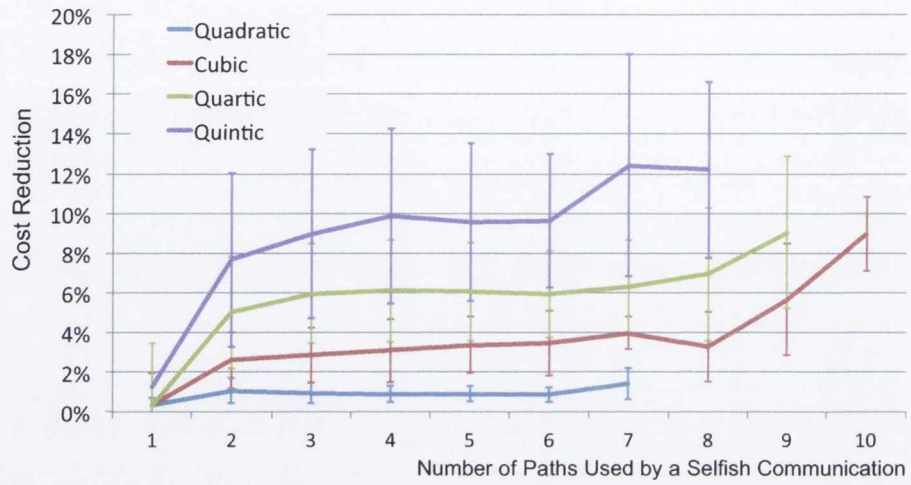
networks for the purpose of fixing some parameters while evaluating the rest. All nodes have the same latency function, which is polynomial and varies from degree 2 (quadratic functions) to degree 5 (quintic functions) in different runs. All nodes also have the same transmission/interference range, which may vary between 4 neighbours and 8 neighbours in different runs. The results are shown in Fig. 6.3.

The first observation we can draw from the results is that the steepness of the latency function plays a positive effect on the cost reduction. In wired networks, it has been demonstrated that the steepness of the latency function affects the analytical low bounds of cost reduction. In wireless networks, we have shown that the lower bound of cost reduction is always 100% as long as the latency function meets certain criteria. Results in Fig. 6.3 demonstrate that although they have the same reduction in best cases, a "steeper" latency function enjoys a higher cost reduction of optimal solutions.

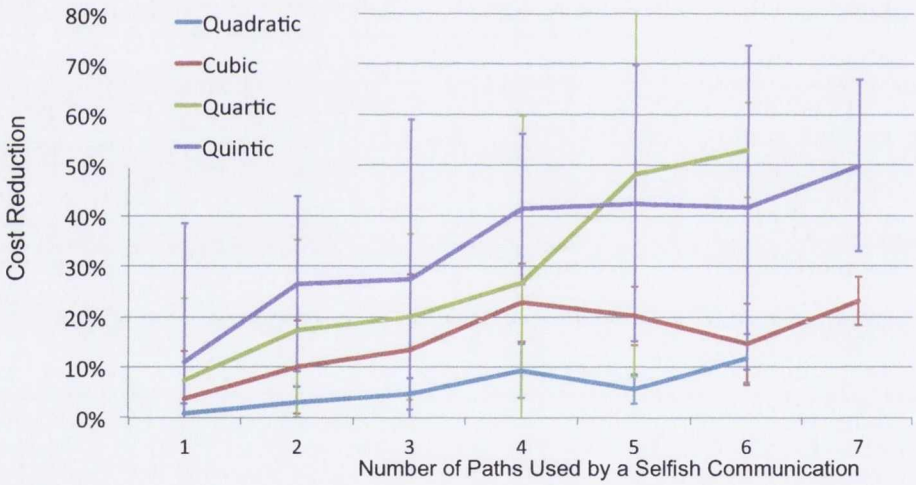
Comparing Fig. 6.3(b) and Fig. 6.3(a), it can be seen that the optimal solutions produced by QBIP reduces more costs from selfish solutions as the number of interferences at each hop increase.

Last but not the least, the average cost reduction increases as the number of paths used by selfish approaches increases. Depending on the topology and the traffic scenario of a network, a communication may utilize various number of paths. Selfish approaches are more likely to be inefficient as the number of paths used increases, which gives more space for optimal approaches to optimize. Combining this observation with the fact that there can be a large amount of potential paths for a communication, one may be curious about the number of paths used by either ideal selfish routing or optimal routing. The histogram of the number of paths used by either solution is summarized in Fig. 6.4.

As shown in Fig. 6.4(a), the majority of communications utilize $2-4$ paths under the 4 neighbour interference range in the 5×5 mesh network. At most, a communication uses 11 paths. This is interesting yet not surprising. Despite the large amount of potential paths, any of them heavily interfere with each other, thus are not utilized. As the number of interferences increases in a fixed size network, the number of paths used, both the

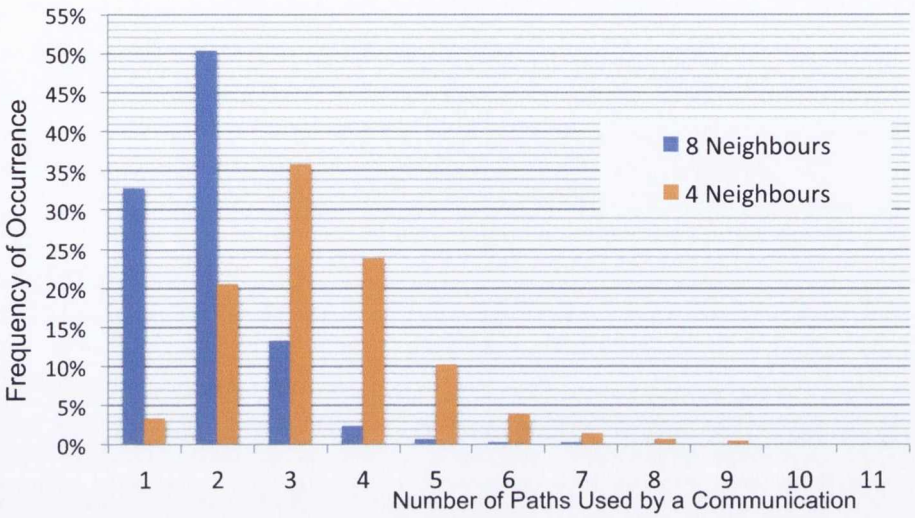


(a) 4 Neighbour Interference Range

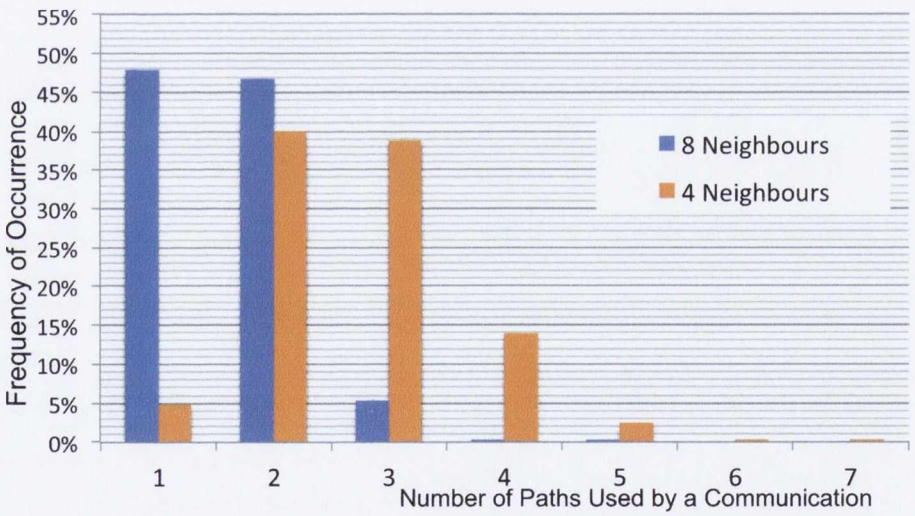


(b) 8 Neighbour Interference Range

Fig. 6.3: Cost Reductions of Different Polynomial Functions



(a) Number of Path Used by Ideal Selfish Routing



(b) Number of Path Used by QBIP

Fig. 6.4: Histogram of Number of Paths Used by Communications

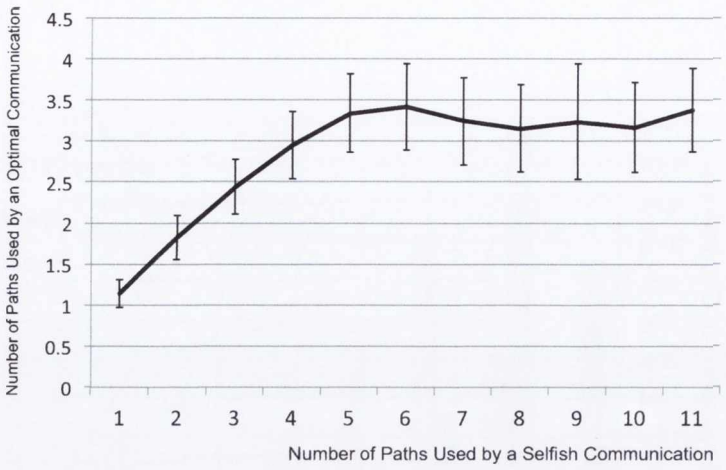


Fig. 6.5: Correlation of Path Used by Optimal and Selfish Approaches

average and the maximum, reduces accordingly.

As shown in Fig. 6.4(b), the number of paths used by an optimal communication lies largely between 2 – 3 paths under the 4 neighbour interference range and between 1 – 2 paths under 8 neighbour interference range. We have observed a maximum of 7 paths used by a communication. A comparison between Fig. 6.4(b) and Fig. 6.4(a) may suggest that optimal routing solutions use on average less paths compared to their ideally selfish counterparts. To better illustrate this observation, we plot the relation between the two value in Fig. 6.5. It can be seen that, for communications whose selfish solutions use single path, their optimal solutions may not always use single paths. For multipath selfish routing solutions, their optimal counterparts are expected to use fewer paths. In addition, the number of paths used by optimal solutions remains stable at 3 – 4 as the number of paths used by selfish approaches increases.

In summary of the first set of experiments, we have learnt the followings: 1) The steepness of latency functions and the number of interferences of each hop impose a positive effect to the cost reduction of optimal routing solutions. 2) The number of paths used by optimal communications is usually less than that used by selfish communications.

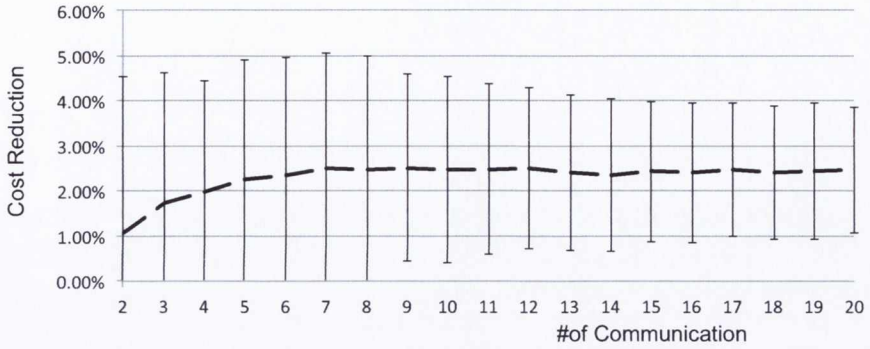


Fig. 6.6: Mean Cost Reduction Over the Number of Interfering Communications

6.2.4 Factors of Cost Reduction - Random Topology

Our second set of experiments are conducted in randomly generated networks. The size of each network varies between 3 – 3000 nodes, with expected node density between 1 – 20 neighbours per hop. The number of communications varies between 2 – 20. The latency function used in these experiments is of a quadratic form, more specifically, $\ell(f, Na) = (f + N_a)^2 + (f + N_a)$

As shown in Fig. 6.6, the average cost reduction of optimal solutions remains roughly stable as the number communication increases. It can be explained by the fact that when a communication makes routing decisions, be it optimal or selfish, it does not differentiate interfering traffic of different sources. Therefore, the number of interfering communications does not affect the cost reduction of optimal solutions[2].

Next, Fig. 6.7 demonstrates that the average length of paths used by a communication plays a negative factor to the cost reduction of optimal solutions. Our interpretation of this result is that as the distance between a source node and a destination node increases, it is more difficult for optimal routing to discover a path with no or little interferences which can largely improve the performance.

[2]It should be noted that, if each communication holds a fixed amount of traffic demand, as the number of communications increases in a network, the cost reduction is expected to change in practise. We will study this effect in queueing networks in subsection 6.2.5. The usage of the quadratic latency function allows the results to be immune to the network traffic demands.

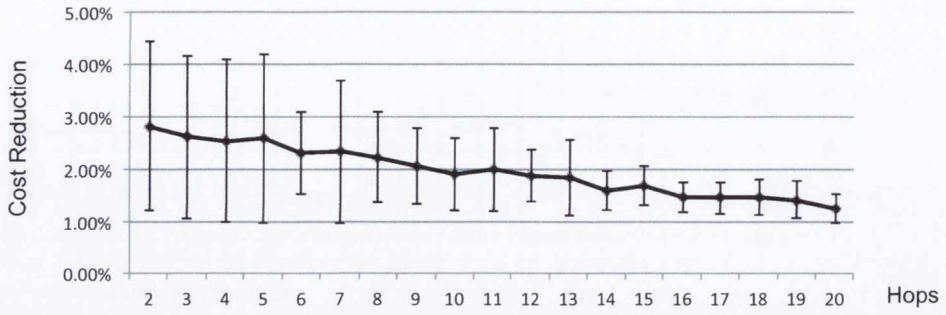


Fig. 6.7: Mean Cost Reduction Over the Lengths of Paths Used.

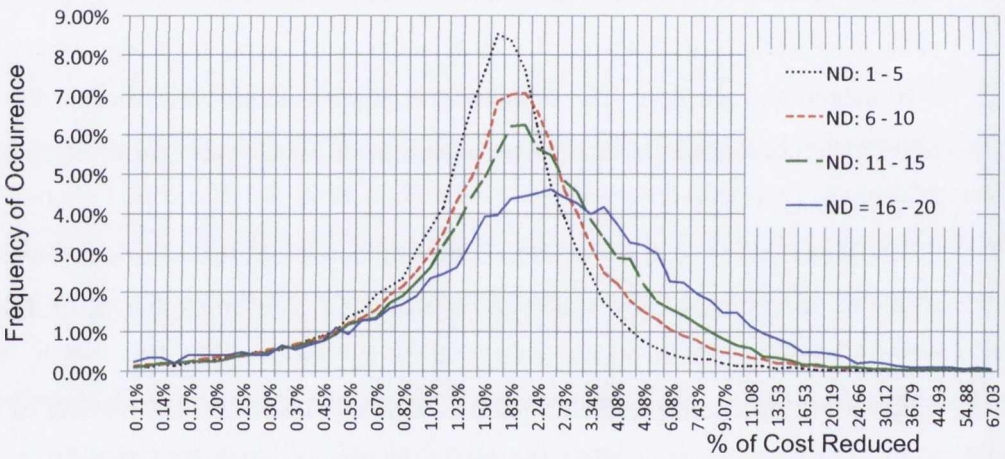


Fig. 6.8: Histogram of Cost Reduction at Different Network Density

	Path Lengths	Number of Paths Used	Degree	Node Density
p-value	6.5445 e-23	9.2735 e-61	0	3.9124 e-204

Table 6.1: Statistical Significance of Experiments: p-value of one-way ANOVA test over different factors

Finally, we summarize results of all experiments in random settings and plot logarithm-transformed histograms of cost reduction for different node densities, as shown in Fig. 6.8. As the node density grows, the average cost reduction of optimal solutions increases. The maximum reduction that we have observed is around 2/3. This result complies with the comparison between 4 neighbour interference range and 8 neighbour interference range, at a larger scale.

In summary of the second set of experiments, we have demonstrated that the average cost reduction is dependent on certain topological features, despite that its analytical upper bound is independent of topologies. The statistical significance of these experiments are illustrated in Table. 6.1. These experiments show that the higher node density and/or the shorter paths used in a network, the larger cost reduction can be expected from optimal solutions. The number of communication flows itself, however, does not affect the cost reduction.

6.2.5 Cost and Capacity

By far, we have discussed experiments using polynomial latency functions under various network settings in order to evaluate the contributing factors of cost reduction. In this subsection, we focus on again the small scale 5×5 mesh network with two communication flows. This time, we aim to evaluate the routing performance with a more realistic latency function at each hop, that is, the M/M/1 queueing delay. Queueing delay differs polynomial functions in that as the traffic of a hop approaches capacity, the delay approaches infinity. Therefore, we can study more accurate capacity-related behaviours of optimal and selfish solutions.

We set the transmission/interference range to cover 4 neighbours of a node. Each hop is assigned a capacity value. At each run, two pair of sources and destinations are randomly selected. Route discovery process is used to find all potential paths. Source nodes compute a selfish solution with traffic demand set as 10% of link capacity, denoted by $\mathcal{D}^{(0)}$. Once the iterative selfish algorithm converges and terminates, the traffic

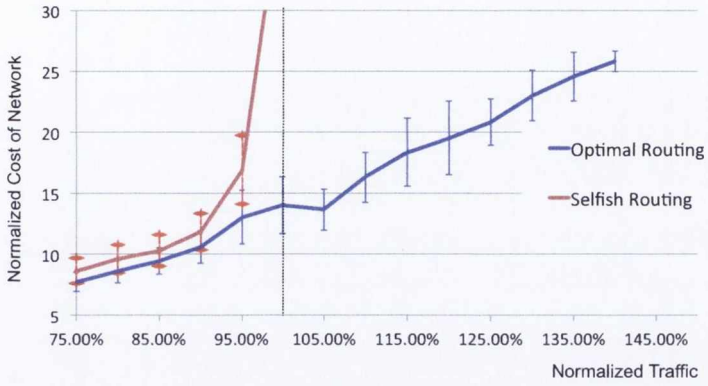
demand is increased by a small portion, denoted by $\Delta\mathcal{D}$. Each source node performs a feasibility check to determine whether the capacity residual of the network can carry the demand increase $\Delta\mathcal{D}$, given the current traffic pattern. A new selfish solution is computed for traffic demand $\mathcal{D}^{(t+1)} = \mathcal{D}^{(t)} + \Delta\mathcal{D}$, t = 0,1..., if the increase of demand is feasible. Otherwise, the algorithm terminates. Then, starting over from $\mathcal{D}^{(0)}$, source nodes compute the optimal solution and gradually increase traffic demands following the same manner until infeasible. This completes one run of our simulations.

We carried out 20 runs of random simulations. Each run produces one sequence of traffic demand and its corresponding cost – $\{\mathcal{D}^{(t)}, C^{(t)}_{selfish}\}$ – for selfish solutions and one sequence $\{\mathcal{D}^{(t)}, C^{(t)}_{optimal}\}$ for optimal solutions. Define $T_{selfish}$ and $T_{optimal}$ as the number of elements in sequences $\{\mathcal{D}^{(t)}, C^{(t)}_{selfish}\}$ for selfish solutions and in $\{\mathcal{D}^{(t)}, C^{(t)}_{optimal}\}$ for optimal solutions respectively. In order to aggregate data of different runs. We normalize results of each run as follows:

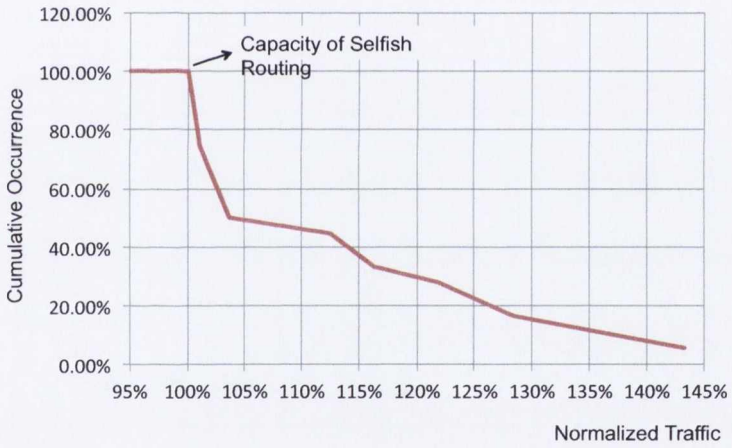
1. Transform the absolute costs into relative costs with respect to the starting optimal costs, i.e. $\tilde{C}^{(t)}_{selfish} = C^{(t)}_{selfish}/C^{(0)}_{optimal}$, $\tilde{C}^{(t)}_{optimal} = C^{(t)}_{optimal}/C^{(0)}_{optimal}$. The reason we choose $C^{(0)}_{optimal}$ as the benchmark is that it is the smallest cost of both series.

2. Transform sequences of traffic demands into relative traffic with respect to the capacity of selfish solutions, i.e. $\tilde{\mathcal{D}}^{(t)} = \mathcal{D}^{(t)}/\mathcal{D}^{(T_{selfish})}$.

The aggregated normalized results are illustrated in Fig. 6.9(a). As traffic of selfish solutions approach their capacities, the delay experienced increases to infinity due to congested queues at intermediate nodes. Optimal solutions enjoy a smaller cost compared to selfish solutions of the same traffic, as expected. This cost reduction on average increases to infinity large as the amount of traffic approaches to selfish capacity. It should be noted that the graph does not include points at the exact capacity, as both selfish and optimal solutions terminate when capacity is reached or exceeded. Instead, points that a fraction of traffic less than capacity are plotted. Around 20% of optimal solutions terminate at the same amount of traffic as selfish solutions. This explains the



(a) Costs of both optimal and selfish solutions are normalized by the starting cost of optimal solutions; Network traffic is normalized by the capacity of selfish solutions



(b) Complementary Cumulative Distribution of Capacity Improvements

Fig. 6.9: Cost and Capacity: In M/M/1 queueing networks, QBIP reduces delay costs of all traffic while improving the throughput of a network

sudden rise of optimal costs (hump of the blue line) near 100% of selfish capacity.

It is interesting to observe that there are about 80% of optimal solutions exceed the capacity of selfish solutions. The complementary cumulative distribution of capacity achieved by optimal solutions is shown in Fig. 6.9(b). Among our 20 runs of simulations, the maximum capacity achieved by optimal solution is more than 43% larger than that of selfish solutions. This result is curious for two reasons:

1. Latency rather than throughput is the objective of our routing algorithm. Yet, it improves the throughput capacity.

2. On the other hand, throughput maximization approaches and their general form – utility maximization approaches – may introduce large latency as they tend to transmit data over long paths.

This improvement of throughput is because in queueing networks, congestions is the main source of cost incurred. Optimal approaches avoid congested bottlenecks, thus improve maximum throughputs.

6.2.6 Summary

In this section, we have focused on the comparison between optimal solutions and selfish solutions. We have demonstrated that in a queueing network, optimal solutions incur less costs than selfish solutions of the same traffic and may improve network capacity. We have observed that the steepness of latency functions, the number of paths used by selfish solutions, node density, and in queueing networks, traffic demands impose positive effects to the cost reduction of optimal solutions. On the other hand, the average lengths of paths plays a negative effect to the cost reduction.

In the next section, we will investigate the convergence of the QBIP algorithm.

6.3 Statistical Evaluation of Convergence

In the previous section, we have studied the comparison between optimal solutions and selfish solutions. In this section, we study the convergence of the proposed QBIP algorithm. The algorithm includes primarily two parts: 1) The quota-based mechanism that addresses coupled constraints in routing problems. 2) The diagonal approximation of the Hessian matrix, which decouples the objective function therefore facilitate the flow-level and node-level distributed manner of the algorithm. All the experiments in this section are conducted in MATLAB [MATLAB, 2012], due to their computation-heavy nature.

We truncate all plots of convergence to show the first 150 steps, even if an algorithm has not yet terminated. The reasons are as follows: A) Even we have assumed persistent traffic demands, 150 rounds is a sufficiently large number, especially given that each round consists the transmission of multiple data packets; B) Unlike other domain where only the terminated solution matters, an end-to-end communication is an ongoing task starting from round 0. Therefore, the early stage of convergence, where cost reduction at each step is drastic, should be the focus.

6.3.1 Quota-based Approach in Decoupling Constraints

In this subsection, we aim to evaluate the performance of the quota-based mechanism in decoupling capacity constraints. The challenge is, we cannot evaluate solely the quota-based mechanism in optimal routing problems because these problems have coupled objective functions. If we use the diagonal approximation of the Hessian matrix to decouple the objective functions, the performance is affected by such an approximation.

In order to tackle this issue, we evaluate the quota-based mechanism in NUM problems. Although NUM problems are not the focus of this thesis, by default their objective functions are decoupled. We set random networks with 500 expected links, 100 single path communications, and an expectation of 20 hops per paths.

In these random networks, we compare the quota-based mechanism against several

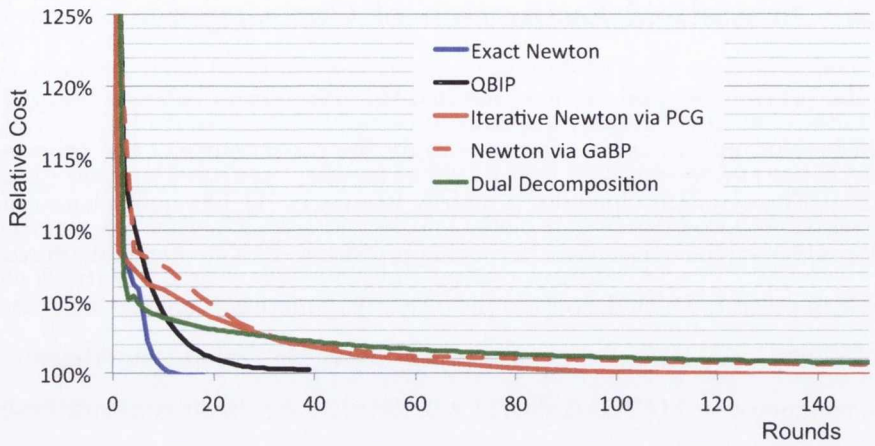


Fig. 6.10: Convergence of Various Optimization Approaches for NUM Problems: Exact Newton and iterative Newton via PCG are centralized approaches; while QBIP, Newton via GaBP and dual decomposition method are flow-level distributed.

existing approaches. As shown in Fig .6.10, the fast converging algorithm is the centralized exact Newton's method, which converges within less 20 steps. However, each Newton step involves the computation of a linear equation system, which can be computationally challenging. Iterative Newton's method can be used to address this problem. We evaluate two types of methods to compute the linear system. In our experiment networks, which are non-sparse, iterative Newton via PCG converges slightly faster as it approaches to the optimal solution. However, this method is centralized. In contrary, Newton via GaBP is a distributed approach, but converges slightly slower in dense networks. The dual decomposition method is a standard distributed technique. It converges slower compared to the rest of the approaches as solutions approach the optimum. The QBIP method converges within 40 steps. It exhibits a superior convergence rate compared to both the iterative Newton methods and the dual decomposition method. This result suggests that the quota-based mechanism alone is an efficient approach to address coupled constraints.

6.3.2 Implementation and Experiment Settings

In the previous subsection, we have demonstrated the efficiency of the quota-based mechanism. In the rest of this section, we will focus on the overall performance of the QBIP algorithm in solving routing optimization problems, which is determined by both the quota-based mechanism and the diagonal approximation to the Hessian matrix. We reiterate the mathematical formulation of routing optimization problems as follows:

$$\begin{aligned} \underset{\mathcal{F}}{\text{minimize}} \quad & C(\mathcal{F}) \\ \text{subject to} \quad & \mathcal{R}\mathcal{F} < \mathcal{C} \\ & \mathcal{F} \succeq 0 \\ & A\mathcal{F} = \mathcal{T} \end{aligned}$$

where $\mathcal{R} \in \mathbb{R}^{m \times n}$ is the routing matrix, $A \in \mathbb{R}^{p \times n}$ and $\mathcal{T} = \{\mathcal{T}_1, ..., \mathcal{T}_p\}^T$ are path-communication matrix and column vector of traffic demand of all communications. Without loss of generality, we choose a quadratic form for cost function because it is a convex increasing function which requires little computational effort of the simulator.

We have implemented a dual decomposition method to address the routing problem. The implementation follows the algorithm [Tan et al., 2006] that we have reviewed in 3.5.2.3, which introduces auxiliary variables to transform the couplings of the objective function to constraints. We have implemented an iterative Newton method using PCG solver included in MATLAB library.

Another iterative method, the GaBP [Bickson, 2009] requires the Hessian matrix to be diagonally dominant, which does not stand in optimal routing problems in general. Although a technique has been proposed to remove such a condition [Johnson et al., 2009], it introduces extra iterations to guarantee the convergence of GaBP. In our simulations, we fail to observe superior results of fixed convergent GaBP compared to the plain dual decomposition approach. Therefore, we will not include the results of fixed convergent GaBP in the rest of this section.

In the next subsection, we will set up a small scale example network to investigate preliminarily the performance of the iterative Newton method, the dual decomposition method and the QBIP algorithm, before moving to large scale random networks in subsection 6.3.4

6.3.3 Preliminary Experiment in a Small Scale Network

We arbitrarily set up a scenario with 3 communications, each of which has two paths. The routing matrix $\mathcal{R}$ is as follows.

$$\mathcal{R} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}^T \qquad A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \tag{6.3}$$

Link capacity is randomly assigned for each link. Traffic demands of all communications are so selected that they sum to 70%[3] of network capacity.

As shown in Fig. 6.11, the QBIP algorithm converges quickly to network optimum. As a matter of fact, it converges to a small neighbourhood of the optimal solution faster than the centralized iterative Newton's method via PCG. Both approaches terminate within $10 - 15$ steps.

On the other hand, the dual decomposition method converges surprisingly slow. Our explanation for this behaviour is the inefficiency of the dual decomposition method to address coupled objective functions. For each link, wireless or wired, where n_l amount of path flows share interference, $n_l^2 - n_l$ amount of auxiliary variables and $n_l(n_l - 1)$ amount of dual variables are introduced. It creates a large number of variables and more importantly, multiple copies of the same variables. In addition, the subgradient

[3]We observe that the percentage does not affect much of results unless it approaches 0% or 100%

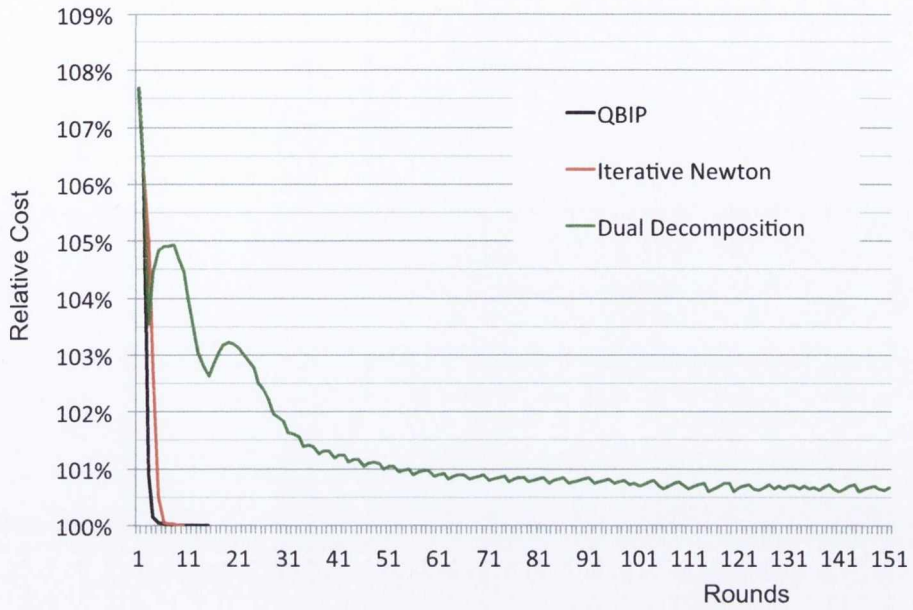


Fig. 6.11: Inefficiency of Dual Decomposition in Routing: Convergence comparison for problem defined by matrices (6.3)

method used to update dual variables is in general slow. Since it is not a descent method, the cost may increase during convergence, as shown in the graph.

Because the dual decomposition method does not exhibit a reasonable convergence in small scale scenarios, we will not evaluate it in the following large scale experiments.

6.3.4 Large Scale Random Experiments

In the following, we focus on evaluations of the convergence of QBIP in addressing routing optimization problems. We choose the centralized iterative Newton's method via PCG as the benchmark algorithm – the control. Routing matrix $\mathcal{R}$ and A are generated randomly within the range specified in Table 6.2.

Fig. 6.12 shows the aggregated convergence of the QBIP and the iterative Newton over various number of interfering communications. The node-level distributed QBIP

Parameters	Range
Number of Links	100 – 2500
Number of Communications	2 – 50
Expected Paths per Communication	2 – 50
Average Length per Path	2 – 20
Number of Interferences per Link	1 – 10

Table 6.2: Parameters in Random Experiments

algorithm exhibits a much faster convergence to a small neighbourhood of the optimal solution compared to the centralized iterative Newton method. This result resembles what observed in the small scale example scenario, Fig. 6.11. This is because each Newton step in the QBIP can be computed analytically, whereas each Newton step is computed by an iterative PCG method.

As the number of interfering communications increases, the convergence is slowed down for both methods. That is, the number of communications is a negative factor for convergence. Nonetheless, it can be seen that the effect of this negative factor grows slowly. The difference between the convergence pattern of 20 and 50 communications is much smaller than that of 2 and 5 communications. This is also true for both methods.

A similar effect can be observed for the expected number of paths per communication, as shown in Fig. 6.13. As the number of paths per communication increases, the convergence of both algorithms slow down.

Finally, the convergence over the number of interfering path flows at each hop is shown in Fig. 6.14. This factor essentially describes the flow density over one hop, which differs from the node density or the amount of traffic of the hop. The number of interferences is a negative factor of convergence.

The usage of diagonal approximation decouples the objective function and facilitates the node-level distributed manner of the QBIP algorithm. However, because of this

approximation, the QBIP algorithm experiences a turning point within a certain neighbourhood of the optimal solution. The convergence slows down quickly after the turning point, although costs are still strictly decreasing at each step.

6.4 Summary

In this chapter, we have evaluate the basic performance of QBIP algorithm. We have compared solutions of QBIP against that of ideal selfish algorithms. In a queueing network, we have demonstrated that QBIP reduces the network costs as well as improving the network capacity.

QBIP exhibits a much superior performance compared to other distributed algorithms, such as the dual decomposition method and the GaBP based Newton method, in both NUM problems and routing problems. Instead, we use centralised approach the iterative Newton via PCG. Across all the scenarios of our simulations, we observed that the QBIP converges faster than iterative Newton to a small neighbourhood of the optimal solutions. Then convergence of QBIP drops to a small pace. The neighbourhood is in general 0.1% - 1% but may vary in different settings.

We argue that the performance of QBIP is desirable compared to iterative Newton, in addition to its node-level distributed manner. Although we have assume persistent traffic, the traffic demand may vary overtime. Also, topologies of ad hoc networks are subject to changes. A fast converging algorithm adapts these dynamism better. In addition, for an iterative optimal routing algorithm, every round of traffic pattern is registered in the performance of algorithm. For example, suppose the cost at each round of algorithm A is $\{10, 3, 2\}$ and that of algorithm B is $\{10, 8, 1\}$. It is apparent that algorithm A is preferable in routing domain, although in other domain B may be preferable. Therefore, it is important for an optimal algorithm to reduce network costs fast in the early stage.

The summary of different optimal algorithms are listed in table 6.3.

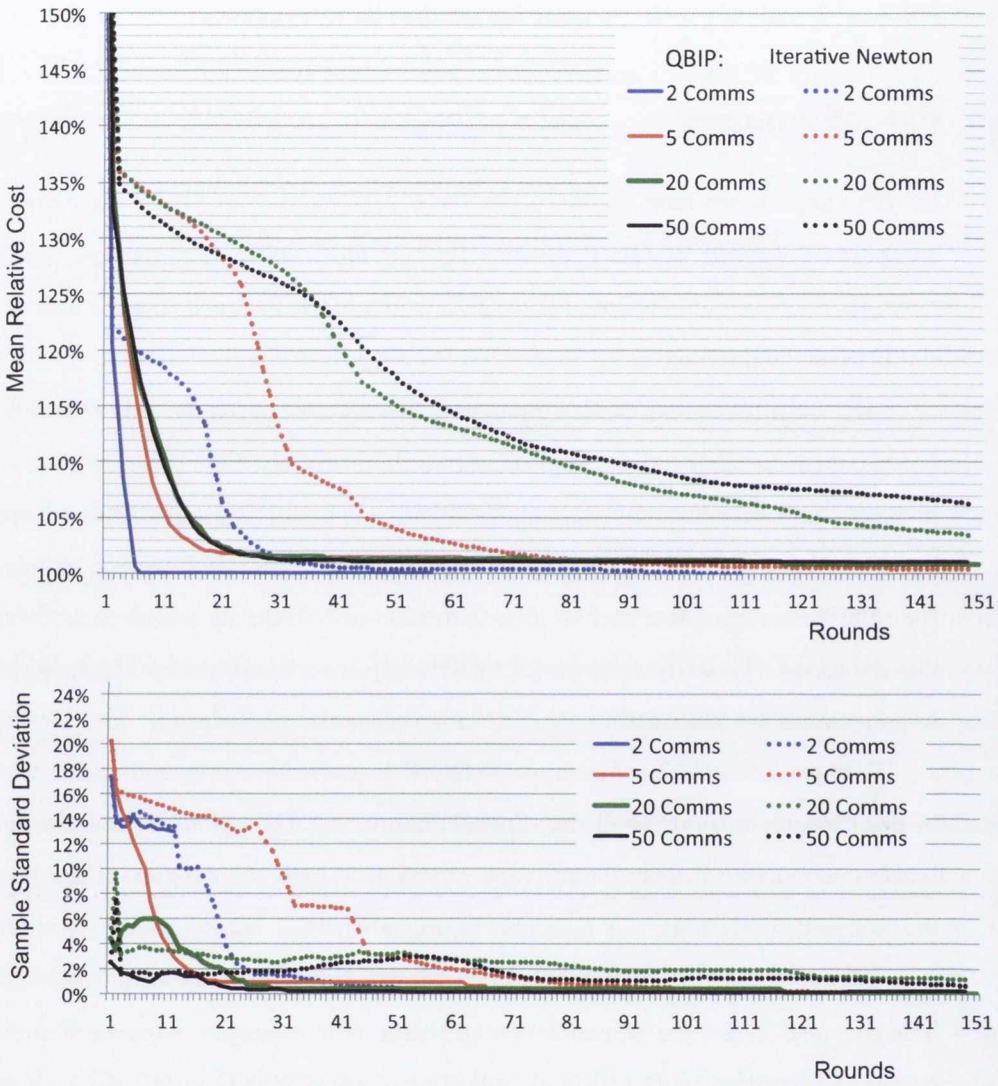


Fig. 6.12: Convergence of QBIP and An Iterative Newton Method: Over Expected Number of Interfering Communications in a Network

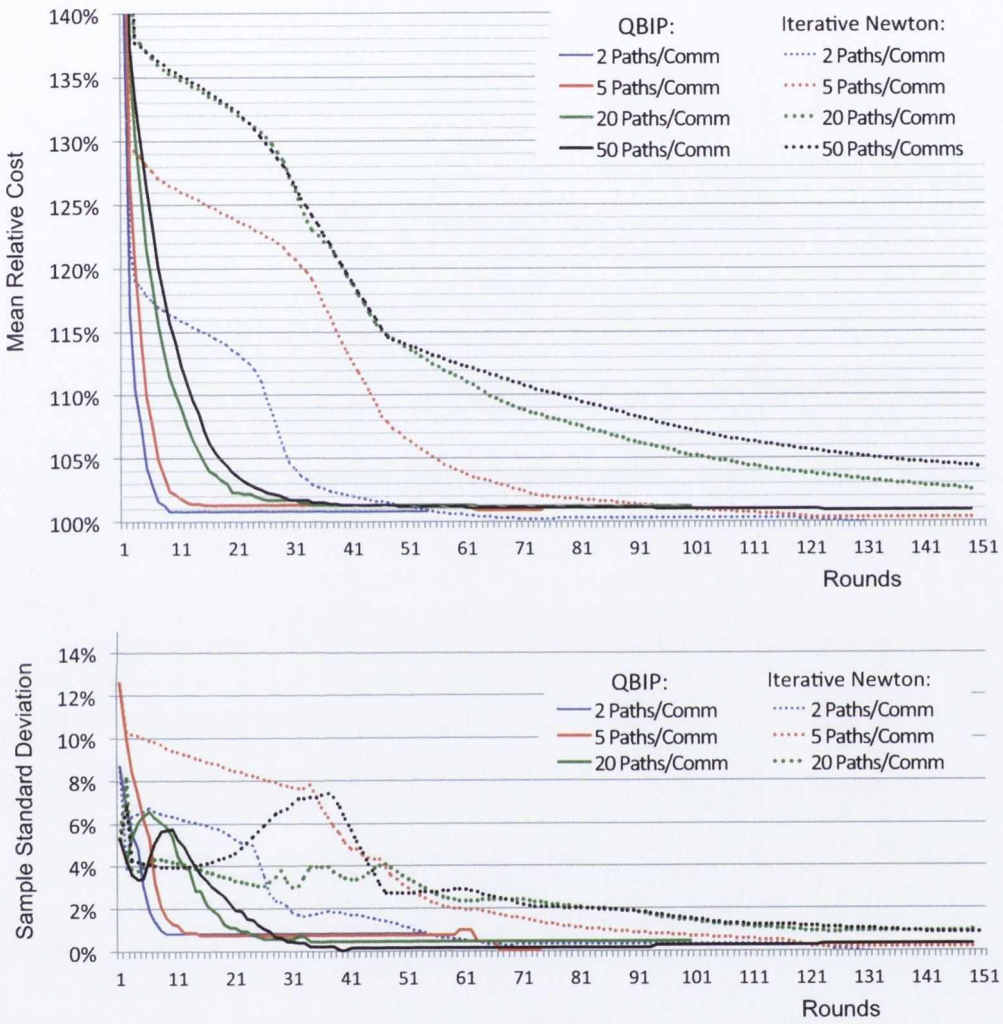


Fig. 6.13: Convergence of QBIP and An Iterative Newton Method: Over Expected Number of Paths Per Communication

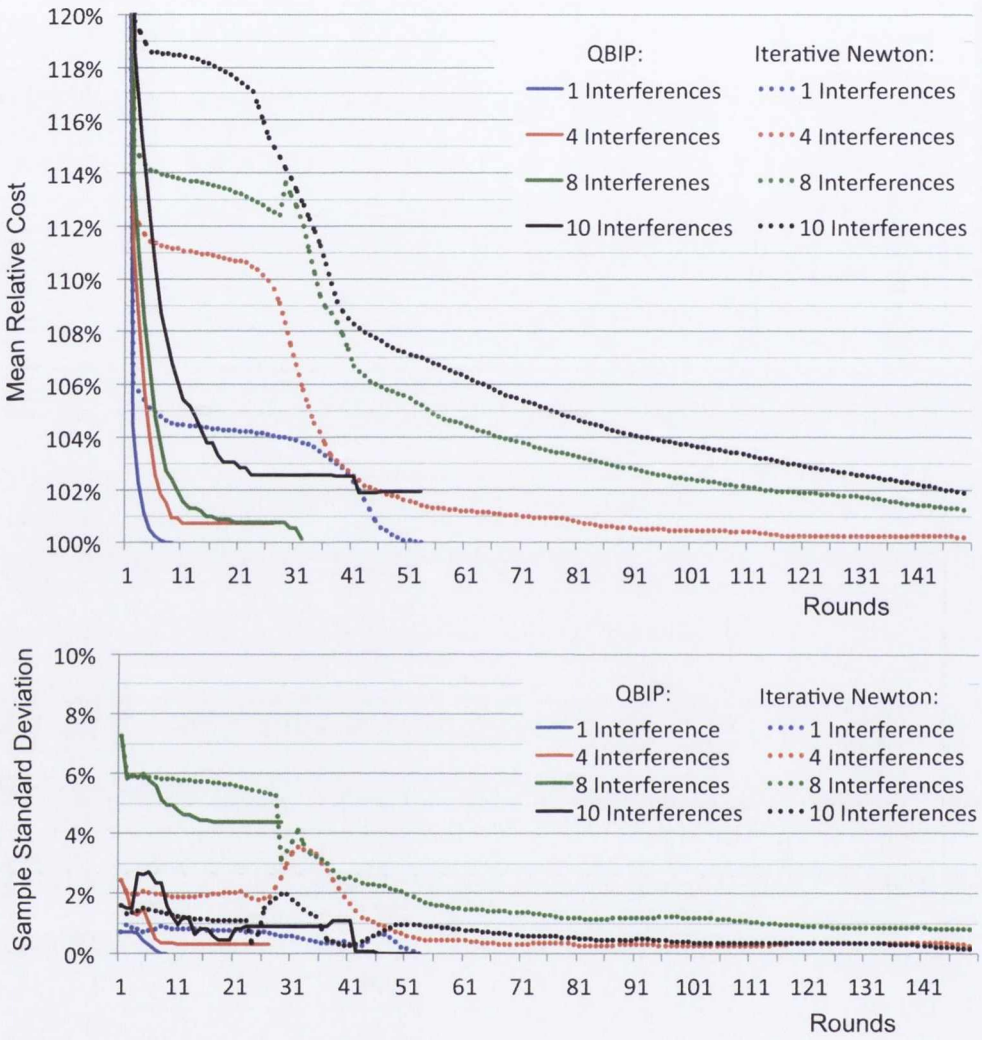


Fig. 6.14: Convergence of QBIP and An Iterative Newton Method: Over Expected Number of Interfering Flows at Each Wireless Hop

Methods	Pros	Cons
QBIP	1) Fast convergence to neighbourhood of optimum 2) Node-level distributed manner with exact same convergence as flow-level distributed manner 3) Integrated with route discovery	The neighbourhood is in general small but may increase in certain scenarios
Dual Decomposition	1) Easy to implement 2) Practical implication – dual variables as price of resources 3) Naturally a flow-level distributed algorithm; can achieve node-level distributed manner by using link rate as handle variables	1) Can be very slow especially in optimal routing problems 2) Not strictly descent due to the usage of subgradient method 3) Node-level distributed manner introduces a large number of variables
Newton via GaBP	1) If it converges, it converges fast, especially if the network is sparse 2) Naturally a flow-level distributed algorithm; can achieve node-level distributed manner by using link rate as handle variables	1) Node-level distributed manner introduces a large number of variables 2) Requires diagonal dominant Hessian in order to converge, which does not stand in optimal routing problems but NUM problems
Iterative Newton via PCG	It converges reasonably fast to network optimum while avoids the computation complexity of Newton	It is centralized thus can not be applied to distributed optimal routing algorithms

Table 6.3: Pros and Cons of Optimization Approaches

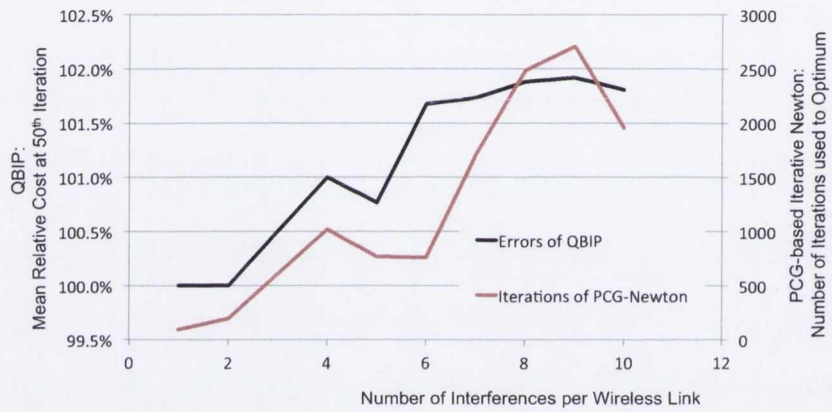


Fig. 6.15: Effects of Increased Local Interferences: The correlation between the convergence of QBIP and that of the iterative Newton's method

Chapter 7

Conclusion

We conclude by reviewing the contributions of this thesis and by describing the future work. In section 7.1, we summarize the research questions that we have identified through the state of the art review. Each of the research questions are addressed by some of our designed approaches. We briefly reiterate our approaches. In section 7.2, we describe our future work following this thesis.

7.1 Summary and Contributions

The objective of this thesis is to investigate the optimal routing problem in wireless ad hoc networks. We aim to connect the domain of wireless communication algorithms with the optimization theory. In chapter 2, we have reviewed the state of the art results in the domain of classical wireless routing algorithms, which focus mainly on the engineering aspect of routing and can not reach or approach optimal solutions. In chapter 3, we have described the background of optimization theory and reviewed some of the notable optimal algorithms that can be applied to wireless routing domain. In section 3.6, we summarized results in the domain of wireless modelling. To date, we have seen few efforts that combine the developments of the three domains.

Through the state of the art review, we have concluded several open questions to be

answered in each domain, in order to achieve wireless routing optimization. The main contribution of this thesis is the solutions to these open questions. In the following, we list the open questions of each domain, followed by our design to answer that question.

7.1.1 Open Questions and Solutions in Optimization Domain

In theoretical optimization domain, these questions are as follows:

1) How can we distributively address the coupled constraints in such way that it coordinates different components quickly and guarantees strict feasibility?

We propose a quota-based mechanism that addresses capacity-constraints with the interior-point method. We introduce auxiliary variables called quotas at each node. It represents the maximum traffic allowed of a path flow at a node. Each node optimizes the quota-assignment locally using second-order Newton's method. Quotas are forwarded to sources of communications, where a second-order interior-point step is computed. The quota-based mechanism guarantees second-order update of solutions, while the standard approach – the dual decomposition method – is bottlenecked by the first-order subgradient update of dual variables. Because logarithm barrier function is used, the quota-based mechanism guarantees strict feasibility. The performance of the quota-based mechanism is verified in NUM experiments where the objective functions are not coupled.

For non-negative constraints, we propose to use a projection method due to the fact the network optimum is an equilibrium of derivative costs of each utilized path.

2) How can we address the coupling introduced by the objective function without using iterative methods at each Newton step?

We use the diagonal approximation of the Hessian matrix in Newton step method. In this way, the computation of each Newton step is distributed with respect to each component and the computational effort required is trivial. Because no iterative methods is need at each Newton step, the algorithm converges fast – similar to centralized exact Newton's method as we have observed in simulations – in the early stage of convergence. The convergence slows down when the solution enters a neighbourhood of the optimal

solution. Such a neighbourhood is varies in different scenarios, as we have demonstrated in section 6.3.4, but is in general small.

Our argument for the usage of diagonal approximation is that , in wireless routing domain, it is crucial to reduce costs quickly in the early stage of convergences. Also, it is reasonable for an algorithm to terminate between 1% and 0.1% away from optimum rather than between $10^{-4}\%$ and $10^{-6}\%$ away from optimum if the latter requires significantly larger amount of communicational and computational overheads.

3) Is it possible to design a node-level distributed algorithm without increasing the size of the optimization problem?

We tackle this problem from a different angle. From the communication and algorithm point of view, paths between two nodes satisfy the optimal substructure property. Therefore instead of breaking the network optimization problem into subproblems at each node as in existing distributed approaches, the flow-level distributed routing optimization can be carried out at each intermediate nodes recursively from destinations to sources. Our approach uses path flows as handle variables. Each node perform part of the flow-level optimization problem. The node-level distributed algorithm achieves the same convergence compared to its flow-level distributed counterparts.

In the contrary, existing node-level distributed algorithms rely on using link flow as handle variables and apply the same mathematical approaches as flow-level distributed algorithms, such as dual decomposition or matrix splitting technique. This approach increases the size of the optimization problem and slows down the convergence compared to flow-level distributed algorithms.

7.1.2 Open Questions and Solutions in Routing Algorithm Domain

Existing optimal routing approaches assume the availability of all potential paths and do no consider route discovery. However current developments route discovery can not be used by optimal routing approaches. This is because each type of route discovery imposes certain path filtering and selection criteria, which may not coincide with opti-

mization methods. In addition, there may be a large amount of potential paths between sources and destinations, which introduces more challenge to the design of route discovery algorithm for optimal routing.

The proposition 4 and its two corollaries establish a path filtering criterion that bridge the optimization approach to route discovery method. It suggests that the dual variable computed as the by-product for each communication can be used as a TTL value to determine whether a path should be filtered or established. In addition, we use the bloom filter to remove loops in the flooding of RREQ messages. Finally, we differentiate the concept of orders of hops in a path and the path length. Based on this observation, we propose to use hop counts to establish orders of intermediate from sources to destinations. In this way, loops can be removed using many existing techniques. At the same, we use derivative costs of paths as their length to determine path filtering and traffic distribution.

7.1.3 Connecting Wireless Domain – the Wireless Medium Cost

The majority of routing algorithms, both classical engineering approaches and optimal approaches, follow the link-model of connected nodes. While it describes properly the topology of wired networks, it can not model the interference between neighbouring wireless transmission.

We classify the wireless interferences into three major categories: primary interference, secondary interference and primary indirect interferences. We propose a wireless medium cost that model the cost incurred due to different wireless interferences. The wireless medium cost is an abstraction of the behaviour at MAC layer, it hides all the complexity and is presented as a function. The derivatives of wireless medium cost coincide with the derivative of wireless costs. We demonstrate the convexity of the wireless medium cost function, which equivalently proves the convexity of wireless cost – a result has not been investigated to the best of our knowledge. Our proposed QBIP algorithm use wireless medium cost function as the objective function in wireless networks.

7.2 Future Work

The future work of this thesis lies also in different domains. In the domain of algorithm design, a more thorough study of the communication overhead of QBIP is desirable. The effect of the bloom filter in reducing the communication overhead is yet to be evaluated.

In the domain of optimization theory, our next step is to fix the error neighbourhood introduced by the diagonal Hessian. A reasonable approach is to apply some of the matrix splitting technique to the QBIP algorithm, which may guarantee a better convergence within the error neighbourhood. Another issue to be addressed is the timely manner of the update. The algorithm at the moment operates in a synchronous manner. Recent results, e.g. [Nedic and Ozdaglar, 2009], suggest potential approaches to translate QBIP to an asynchronous algorithm.

Finally, in the domain of wireless modelling, we have assume that the latency function treat all transmissions homogeneously. That is $\mathcal{M}_{a,i} = \sum_j f_j^{(l)}$ in $\ell_{a,i}^{(s)}(\mathcal{M}_{a,i})$ and $\mathcal{S}_{a,i}^{(2)}(\mathcal{M}_{a,i})$. A more accurate model would be treat $\mathcal{M}_{a,i}$ as a vector instead of a scalar. Results can be computed in a similar way as we demonstrated in appendix C.

Appendix A

Wireless Queueing Delay with Poisson Arrival

In the following, we give the derivation of the queueing delay of a wireless that has multiple outgoing traffic flows to different neighbours. Each of these link flows may experience different interferences, which results in different medium access delay. Assuming we know the mean $\ell_{a,i}^{(a)}$ and the second moment $\mathcal{S}_{a,i}^2$, the proof is as follows.

Proof. The queueing delay of a M/G/1 queue is given as follows [Gross et al., 2008]:

$$\ell_a^{(q)} = \frac{\bar{f}_a \mathcal{S}_a^{(2)}}{2(1-\rho)} \tag{A.1}$$

where $\bar{f}_a$ is the arrival rate of the queue; $\mathcal{S}_a^{(2)}$ is the second moment of the service time; and ρ is the utilization of the queue. It is easily known that $\bar{f}_a = \sum_{f_i^{(l)} \in \mathcal{F}_a} f_i^{(l)}$. The utilization is the sum of utilization for each class of traffic, $\rho = \sum_{i:f_i^{(l)} \in \mathcal{F}_a} \ell_{a,i}^{(s)}(\mathcal{M}_{a,i}) f_i^{(l)}$ [Gross et al., 2008].

To find the second moment of the long-term service time, let us consider the problem within a large time interval $T \gg 0$. We define the duration of t^{th} medium access of flow j as $X_{t,j}$. Let N_j denotes the number of medium accesses for flow j during the interval T. According to the definition of the second moment of discrete random variables, we

have the second moment of service time over the duration T as:

$$\begin{aligned}\mathcal{S}_a^{(2)}[T] =& \frac{\sum_{i:f_i^{(l)}\in\mathcal{F}_a}\sum_{t=1}^{N_j} X_{t,i}^2}{\sum_{i:f_i^{(l)}\in\mathcal{F}_a} N_j} \\ =& \frac{\sum_{i:f_i^{(l)}\in\mathcal{F}_a}\frac{\sum_{t=1}^{N_i} X_{t,i}^2}{N_j} N_i}{\sum_{i:f_i^{(l)}\in\mathcal{F}_a} N_i} \\ =& \frac{\sum_{i:f_i^{(l)}\in\mathcal{F}_a}\mathcal{S}_{a,i}^2(\mathcal{M}_{a,i}) N_i}{\sum_{i:f_i^{(l)}\in\mathcal{F}_a} N_i}\end{aligned}$$

Since we have assumed that the long-term arrival rate is less than the long-term service rate, the long-term expected number of medium accesses is the expected arrivals $Tf_i^{(l)}$ for flow i. Taking T to infinity, we have:

$$\mathcal{S}_a^{(2)} = \frac{\sum_{i:f_i^{(l)}\in\mathcal{F}_a}\mathcal{S}_{a,i}^2(\mathcal{M}_{a,i}) f_i^{(l)} T}{\sum_{i:f_i^{(l)}\in\mathcal{F}_a} f_i^{(l)} T} = \frac{\sum_{i:f_i^{(l)}\in\mathcal{F}_a}\mathcal{S}_{a,i}^2(\mathcal{M}_{a,i}) f_i^{(l)}}{\bar{f}_a} \tag{A.2}$$

Taking (A.2) into the delay of M/G/1 queue, we reach the queueing delay at each node:

$$\ell_a^{(q)} = \frac{\sum_{i:f_i^{(l)}\in\mathcal{F}_a}\mathcal{S}_{a,i}^{(2)}(\mathcal{M}_{a,i}) f_i^{(l)}}{2\left(1-\sum_{f_i^{(l)}\in\mathcal{F}_a}\ell_{a,i}^{(s)}(\mathcal{M}_{a,i}) f_i^{(l)}\right)}$$

□

Appendix B

Proof of Convexity

In the following, we prove that the optimal routing problem is strictly convex under the condition that the medium access delay of the MAC approach used is convex.

B.1 Proof of Lemma 1

Proof. The purpose of this proof is to demonstrate that for any node a and any link flow $f_i^{(l)}, f_j^{(l)} \in \mathcal{F}_a \cup \mathcal{I}_a \subseteq \mathcal{F}^{(l)}$, the second derivatives of queueing delay at node a is strictly positive, namely,

$$\frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial f_i^{(l)} \partial f_j^{(l)}} > 0$$

given the condition that the medium access delay and its second moment are convex.

Firstly, we describe the notation used in this proof. We follow the numerator-layout notation in matrix calculus. In such a layout, each vector is column vector by default. The partial derivative $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$, where $\mathbf{y} \in \mathbb{R}^{m \times 1}$, $\mathbf{x} \in \mathbb{R}^{n \times 1}$ is a $m \times n$ matrix, which is usually referred to as the Jacobian matrix. Without loss of generality, we let $\pi_{a,i} \in \mathbb{R}^{2 \times 1}$ denote a two-dimensional vector, i.e. $\pi_{a,i} = \{f_i^{(l)}, \mathcal{M}_{a,i}\}^T$. Let $n_a = |\mathcal{F}_a|$ denote the number of outgoing link flows of node a. Each π_a consists of n_a amount of $\pi_{a,i}$. According to our formulation, naturally π_a is of a matrix form, that $\pi_a \in \mathbb{R}^{2 \times n_a}$. However, a powerful tool

- the chain rule of differentiation - does not apply to matrix-valued or matrix-argument functions, for example $\ell_a^{(q)}(\cdot)$ in our case. In order to ease the proof, we construct π_a in a vector form: $\pi_a = \{\pi_{a,1}^T, ..., \pi_{a,n_a}^T\}^T \in \mathbb{R}^{2n_a \times 1}$.

Secondly, in order to show that the second derivatives are strictly positive, we need to decompose the second derivative into known terms. In order to do so, we rewrite equation (4.6) into a vector form,

$$\begin{aligned}
\frac{\partial \ell_a^{(q)}(\pi_a)}{\partial f_i^{(l)}} =& \frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_a} \frac{\partial \pi_a}{\partial f_i^{(l)}} \\
=& \frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \{\pi_{a,1}^T, ..., \pi_{a,n_a}^T\}^T} \frac{\partial \{\pi_{a,1}^T, ..., \pi_{a,n_a}^T\}^T}{\partial f_i^{(l)}} \\
=& \{\frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,1}}, ..., \frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,n_a}}\} \{\frac{\partial \pi_{a,1}}{\partial f_i^{(l)}}^T, ..., \frac{\partial \pi_{a,n_a}}{\partial f_i^{(l)}}^T\}^T
\end{aligned} \tag{B.1}$$

Taking one more derivatives to (B.1), the second derivative can be decomposed as follows:

$$\begin{aligned}
\frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial f_i^{(l)} \partial f_j^{(l)}} =& \frac{\partial}{\partial f_j^{(l)}} \Big(\frac{\partial \ell_a^{(q)}(\pi_a)}{\partial f_i^{(l)}} \Big) \\
=& \{\frac{\partial \pi_{a,1}}{\partial f_i^{(l)}}^T, ..., \frac{\partial \pi_{a,n_a}}{\partial f_i^{(l)}}^T\} \frac{\partial}{\partial f_j^{(l)}} \Big(\{\frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,1}}, ..., \frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,n_a}}\}^T \Big) \\
+& \{\frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,1}}, ..., \frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,n_a}}\} \frac{\partial}{\partial f_j^{(l)}} \Big(\{\frac{\partial \pi_{a,1}}{\partial f_i^{(l)}}^T, ..., \frac{\partial \pi_{a,n_a}}{\partial f_i^{(l)}}^T\}^T \Big)
\end{aligned}$$

Since $\frac{\partial \pi_{a,r}}{\partial f_i^{(l)}}, \forall r = 1...n_a$ is a constant, the value $\frac{\partial}{\partial f_j^{(l)}} \Big(\{\frac{\partial \pi_{a,1}}{\partial f_i^{(l)}}^T, ..., \frac{\partial \pi_{a,n_a}}{\partial f_i^{(l)}}^T\}^T \Big)$ is always zero. The above equation reduces to:

$$\begin{aligned}
\frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial f_i^{(l)} \partial f_j^{(l)}} =& \{\frac{\partial \pi_{a,1}}{\partial f_i^{(l)}}^T, ..., \frac{\partial \pi_{a,n_a}}{\partial f_i^{(l)}}^T\} \frac{\partial}{\partial f_j^{(l)}} \Big(\{\frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,1}}, ..., \frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,n_a}}\}^T \Big) \\
=& \{\frac{\partial \pi_{a,1}}{\partial f_i^{(l)}}^T, ..., \frac{\partial \pi_{a,n_a}}{\partial f_i^{(l)}}^T\} \{\frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial f_j^{(l)} \partial \pi_{a,1}}, ..., \frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial f_j^{(l)} \partial \pi_{a,n_a}}\}^T
\end{aligned} \tag{B.2}$$

The second part is a column vector, which can be expanded further. For any $r = 1...n_a$,

$$\begin{aligned}\frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial f_j^{(l)} \partial \pi_{a,r}} =& \frac{\partial}{\partial \pi_{a,r}} \Big(\frac{\partial \ell_a^{(q)}(\pi_a)}{\partial f_j^{(l)}} \Big) \\ =& \{ \frac{\partial \pi_{a,1}}{\partial f_j^{(l)}}^T, ..., \frac{\partial \pi_{a,n_a}}{\partial f_j^{(l)}}^T \} \frac{\partial}{\partial \pi_{a,r}} \Big(\{ \frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,1}}, ..., \frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,n_a}} \}^T \Big) \\ =& \{ \frac{\partial \pi_{a,1}}{\partial f_j^{(l)}}^T, ..., \frac{\partial \pi_{a,n_a}}{\partial f_j^{(l)}}^T \} \{ \frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r} \partial \pi_{a,1}}, ..., \frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r} \partial \pi_{a,n_a}} \}^T \end{aligned} \tag{B.3}$$

Similarly, the second part is a column vector, for any $s = 1...n_a$, $\frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r} \partial \pi_{a,s}}$ is a row vector of four scalar value,

$$\frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r} \partial \pi_{a,s}} = \{ \frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(1) \partial \pi_{a,s}(1)}, \frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(1) \partial \pi_{a,s}(2)}, \frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(2) \partial \pi_{a,s}(1)}, \frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(2) \partial \pi_{a,s}(2)} \} \tag{B.4}$$

Combining (B.2), (B.3) and (B.4), we have the final decomposition of the second derivative:

$$\begin{aligned}\frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial f_i^{(l)} \partial f_j^{(l)}} =& \sum_{r,s=1}^{n_a} \frac{\partial \pi_{a,r}}{\partial f_i^{(l)}}^T \Big(\frac{\partial \pi_{a,s}}{\partial f_j^{(l)}}^T \frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r} \partial \pi_{a,s}}^T \Big) \\ =& \sum_{r,s=1}^{n_a} \Big(\frac{\partial \pi_{a,r}(1)}{\partial f_i^{(l)}} \frac{\partial \pi_{a,s}(1)}{\partial f_j^{(l)}} \frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(1) \partial \pi_{a,s}(1)} + \frac{\partial \pi_{a,r}(1)}{\partial f_i^{(l)}} \frac{\partial \pi_{a,s}(2)}{\partial f_j^{(l)}} \frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(1) \partial \pi_{a,s}(2)} \\ &+ \frac{\partial \pi_{a,r}(2)}{\partial f_i^{(l)}} \frac{\partial \pi_{a,s}(1)}{\partial f_j^{(l)}} \frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(2) \partial \pi_{a,s}(1)} + \frac{\partial \pi_{a,r}(2)}{\partial f_i^{(l)}} \frac{\partial \pi_{a,s}(2)}{\partial f_j^{(l)}} \frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(2) \partial \pi_{a,s}(2)} \Big) \end{aligned} \tag{B.5}$$

In equation (B.5), each of the eight coefficients, e.g. $\frac{\partial \pi_{a,r}(1)}{\partial f_i^{(l)}}$, is either 0 or 1. Our final step of this proof is to demonstrate that for $x = 1, 2 \;\; y = 1, 2$, $\frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(x) \partial \pi_{a,s}(y)}$ is always non-negative. To see this, let us examine equation (4.5). We reiterate it as follows:

$$\begin{cases} \frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(1)} = \frac{\mathcal{S}_{a,r}^{(2)}(\mathcal{M}_{a,r}) B + \ell_{a,r}^{(s)}(\mathcal{M}_{a,r}) \sum_{v: f_v^{(l)} \in \mathcal{F}_a} \mathcal{S}_{a,v}^{(2)}(\mathcal{M}_{a,v}) f_v^{(l)}}{2B^2} \\ \frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(2)} = \frac{\mathcal{S}_{a,r}^{(2)'}(\mathcal{M}_{a,r}) f_r^{(l)} B + \ell_{a,r}^{(s)'}(\mathcal{M}_{a,r}) f_r^{(l)} \sum_{v: f_v^{(l)} \in \mathcal{F}_a} \mathcal{S}_{a,v}^{(2)}(\mathcal{M}_{a,v}) f_v^{(l)}}{2B^2} \end{cases}$$

where $B = 1 - \sum_{f_v^{(l)} \in \mathcal{F}_a} \ell_{a,v}^{(s)}(\mathcal{M}_{a,v}) f_v^{(l)}$. Taking partial derivative $\frac{\partial}{\partial \pi_{a,s}(y)}$ over either sub-equation above gives one of the four cases of $\frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(x) \partial \pi_{a,s}(y)}$. To simplify the process, we

can unify $\frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(1)}$ and $\frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(2)}$ into the following form

$$\frac{\partial \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(x)} = \frac{DB + E}{2B^2}$$

where D and E represents different terms for x =1 and x = 2. With a slight abuse of notations, let D', B' and E' denote the partial derivative of each component with regards to $\pi_{a,s}(y)$ for either y=1 or y=2. Then $\frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(x)\partial \pi_{a,s}(y)}$ has a unified form as follows:

$$\begin{aligned}\frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(x)\partial \pi_{a,s}(y)} =& \frac{(D'B + DB' + E')B^2 - 2(DB + E)BB'}{2B^4} \\ =& \frac{D'B^2 + E'B - B'(DB^2 + 2E)}{2B^3}\end{aligned} \tag{B.6}$$

In either one of the four cases, because $\ell_{a,i}^{(s)}(\cdot)$ and $\mathcal{S}_{a,i}^{(2)}(\cdot)$ are positive convex increasing functions, it is easily known that $D, D', B, E, E' \geq 0$ and $B' \leq 0$. Therefore, $\frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(x)\partial \pi_{a,s}(y)} \geq 0$ at all time.

In addition, it can be verified that for any $f_i^{(l)}, f_j^{(l)} \in \tilde{\mathcal{F}}_a^{(l)} = \mathcal{F}_a \cup \mathcal{I}_a$, there exist $r, s = 1, ..., n_a$ and $x, y = 1, 2$, such that

$$\frac{\partial \pi_{a,r}(x)}{\partial f_i^{(l)}} \frac{\partial \pi_{a,s}(y)}{\partial f_j^{(l)}} \frac{\partial^2 \ell_a^{(q)}(\pi_a)}{\partial \pi_{a,r}(x)\partial \pi_{a,s}(y)} > 0$$

Therefore, the queueing delay $\ell_a^{(q)}(\pi_a) = \tilde{\ell}_a^{(q)}(\tilde{\mathcal{F}}_a^{(l)})$is convex over the set of link flows $\mathcal{F}^{(l)}$ and is strictly convex over $\mathcal{F}_a \cup \mathcal{I}_a$ □

B.2 Proof of Lemma2

Proof. The purpose of this proof is to show that if the queueing delay and the medium access delay is convex then the corresponding costs are strictly convex. According to equation (4.13), both queueing costs and medium access costs can be write in the form of $\tilde{\ell}(\tilde{\mathcal{F}}_a^{(l)})T\tilde{\mathcal{F}}_a^{(l)}$. In the following, we show that the convexity of $\tilde{\ell}(\tilde{\mathcal{F}}_a^{(l)})$ leads to the convexity of $\tilde{C}(\tilde{\mathcal{F}}_a^{(l)}) = \tilde{\ell}(\tilde{\mathcal{F}}_a^{(l)})T\tilde{\mathcal{F}}_a^{(l)}$.

To do so, we use a standard definition of convex function [Boyd and Vandenberghe, 2004]: A function $g(\cdot)$ is convex if for any $x, y \in \mathbf{dom}\ g$, $x \neq y$ and any $\theta \in [0,1]$,

$$g\big(\theta x + (1-\theta)y\big) \leq \theta g(x) + (1-\theta)g(y) \tag{B.7}$$

The inequality is strict for strictly convex $g(\cdot)$

Apply this property to the convex $\tilde{\ell}(\cdot)$ in $\tilde{C}(\tilde{\mathcal{F}}_a^{(l)})$ we have: For any $x, y \in \mathbf{dom}\ \tilde{\ell}(\cdot)$, $x \neq y$

$$\begin{aligned}
&\tilde{C}\big(\theta x + (1-\theta)y\big) = \tilde{\ell}\big(\theta x + (1-\theta)y\big)T\big(\theta x + (1-\theta)y\big) \\
\leq &\theta\tilde{\ell}(x)T\big(\theta x + (1-\theta)y\big) + (1-\theta)\tilde{\ell}(y)T\big(\theta x + (1-\theta)y\big) \\
= &\theta^2\tilde{\ell}(x)Tx + (1-\theta)^2\tilde{\ell}(y)Ty + \theta(1-\theta)\tilde{\ell}(x)Ty + \theta(1-\theta)\tilde{\ell}(y)Tx
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\tilde{C}\big(\theta x + (1-\theta)y\big) - \Big(\theta\tilde{C}(x) + (1-\theta)\tilde{C}(y)\Big) \\
\leq &\theta^2\tilde{\ell}(x)Tx + (1-\theta)^2\tilde{\ell}(y)Ty + \theta(1-\theta)\tilde{\ell}(x)Ty + \theta(1-\theta)\tilde{\ell}(y)Tx \\
&- \theta\tilde{\ell}(x)Tx - (1-\theta)\tilde{\ell}(y)Ty \\
= &\theta(1-\theta)\tilde{\ell}(x)T\big(y-x\big) + \theta(1-\theta)\tilde{\ell}(y)T\big(x-y\big) \\
= &\theta(1-\theta)\Big(\tilde{\ell}(x) - \tilde{\ell}(y)\Big)(y-x) < 0
\end{aligned}$$

Note that the last step is because $\tilde{\ell}(\cdot)$ is monotonically increasing. Assigning $\tilde{\ell}(\cdot)$ with $\tilde{\ell}_a^{(q)}$ and $\tilde{\ell}_{a,i}^{(s)}$, T with T_a and $T_{a,i}$, we conclude that both the medium access delay cost and the queueing delay cost are strictly convex. This completes our proof. □

B.3 Proof of Proposition 2

Proof. With Lemma 1 and 2 established, the proof of Proposition 2 is trivial. Since $\tilde{\mathcal{F}}_a^{(l)}$ is a subset of $\mathcal{F}^{(l)}$, multiplying a scalar coefficient θ to $\mathcal{F}^{(l)}$ results in all $\tilde{\mathcal{F}}_a^{(l)}$ multiplied by θ in equation (4.13). Using the definition of convex function (B.7), the proposition is established. □

Appendix C

A Case Study for Wireless Medium Access Delay

In the following, we briefly describe the search for a closed form medium access delay function, namely $\ell_{a,i}^{(s)}(\mathcal{M}_{a,i})$, for contention-based 802.11 DCF.

Recall that, under the Poisson arrival assumption, Malone et al. [Malone et al., 2007] have developed an advanced analysis of the medium access delay in a heterogeneous non-saturated 802.11 DCF network. Since our study also assume a Poisson arrival, we choose Malone's model to compute the medium access delay.

For an routing algorithm to compute the derivatives as well as second derivatives, a closed form analytical model is needed. However, due to the complex nature of wireless modelling, existing studies present results in a large non-linear equation system, which describes the relations between different inputs, such as parameters, data loads, etc. In order to extract a closed-form latency function, we vary the data traffic input in Malone's model from $f = 0 \rightarrow$ threshold and solve the non-linear system. For each value of the input series $\{f^{(i)}\}$, we compute a value of latency for the output series $\{\ell_{a,i}^{s}(f^{(i)})\}$. We then performs a non-linear regression on the series of discrete points. We discover that

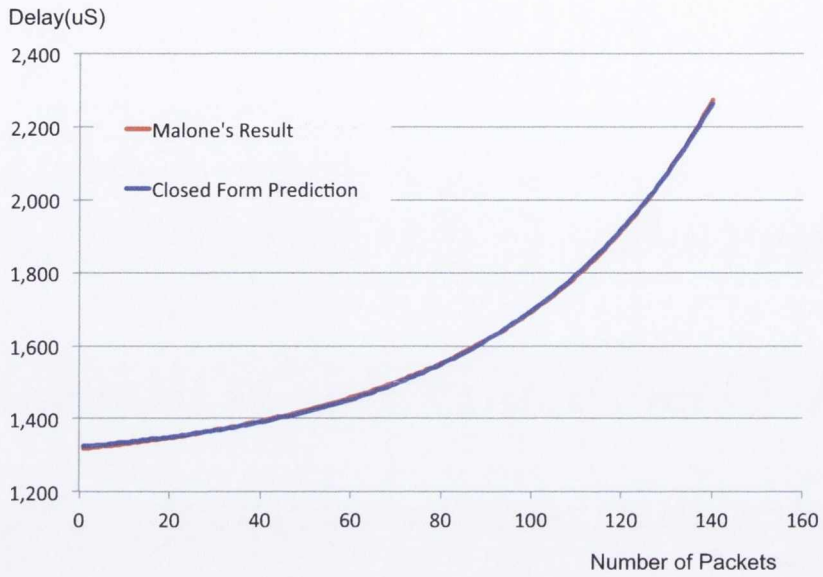


Fig. C.1: Closed Form Delay Function v.s. Malone's Analytical Model

an exponential function fits the discrete points surprisingly well.

$$\ell^{(s)} = b_1 e^{b_2 \mathcal{M}} + b_3$$

For example, the closed-form delay function of a wireless hop with three homogeneous senders is shown in Fig. C.1.

Finally, we move on to vary the number of stations in a wireless hop and discovered that the parameters computed from regression follow a nice linear relation.

In summary, for a 11Mbps 802.11 protocol, with 500 bytes data packet, the medium access delay that we have computed is given as follows:

$$\ell_{a,i}^{(s)}(\mathcal{M}) = (4.07705n + 36.1362)e^{0.008\mathcal{M}} + 23.2423n + 1225.9$$

where n is the number of stations contending for medium access of node a, link i.

Using the same approach, the second moment of medium access delay, $\mathcal{S}^2$, can be calculated in a closed-form as well.

Appendix D

Proof of the Path Filtering Criterion

D.1 Proof of Proposition 4

Now we give the proof of proposition 4

Proof. Because f_{p_j} was not utilized in the previous round, $\Delta f_{p_j} \geq 0$. Assume $\Delta f_{p_j} = 0$ is computed from the appended KKT system.

Since $H = diag(\nabla^2 C)^{-1}$,according to the projected methods,

$\Delta f_{p_j} = max\{-\frac{C'_{\&p_j}+\nu^+}{C''_{\&p_j}}, 0\}$, we have:

$$-\frac{C'_{\&p_j}+\nu^+}{C''_{\&p_j}} \leq 0 \tag{D.1}$$

therefore

$$C'_{\&p_j} \geq -\nu^+ \tag{D.2}$$

From the equality constraint, we know that

$$-\sum_{p_i \in \tilde{\mathcal{P}}_{sd}^{(t)}} \frac{C'_{\&p_i}+\nu^+}{C''_{\&p_i}} - \frac{C'_{\&p_j}+\nu^+}{C''_{\&p_j}} = 0$$

Combine with (D.1), we have

$$\sum_{p_i \in \tilde{\mathcal{P}}_{sd}^{(t)}} \frac{C'_{\&p_i} + \nu^+}{C''_{\&p_i}} \leq 0 \tag{D.3}$$

Recall that the latency function $\ell(\cdot)$ is convex. It can be derived that $C''_{\&p_i} > 0$. Solving (D.3),

$$\nu^+ \leq -\frac{\sum_{p_i \in \tilde{\mathcal{P}}_{sd}^{(t)}} \frac{C'_{\&p_i}}{C''_{\&p_i}} \prod_{p_i \in \tilde{\mathcal{P}}_{sd}^{(t)}} C''_{\&p_i}}{\sum_{p_i \in \tilde{\mathcal{P}}_{sd}^{(t)}} \prod_{i \neq j} C''_{\&p_j}} \tag{D.4}$$

Note that the right side form of (D.4) is essentially $\nu^{(t)}$, computed from the original KKT system, so:

$$\nu^+ \leq \nu^{(t)} \tag{D.5}$$

Combine (D.2) and (D.5),

$$C'_{\&p_j} \geq -\nu^{(t)} \tag{D.6}$$

This completes our proof. □

Bibliography

[IGR, 1991] (1991). An introduction to igrp. White paper 26825, Cisco.

[Albrightson et al., 1994] Albrightson, B., Garcia-Luna-Aceves, J., and Boyle, J. (1994). Eigrp-a fast routing protocol based on distance vectors. In *Proceedings of Networld/Interop.*

[Altman et al., 2008] Altman, E., Bernhard, P., and Silva, A. (2008). The mathematics of routing in massively dense ad-hoc networks. In *Proceedings of the 7th international conference on Ad-hoc, Mobile and Wireless Networks*, pages 122-134, Berlin, Heidelberg.

[Awduche et al., 2001] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and Swallow, G. (2001). RSVP-TE: Extensions to RSVP for LSP Tunnels. RFC 3209 (Proposed Standard). Updated by RFCs 3936, 4420, 4874, 5151, 5420, 5711.

[Basu et al., 2003] Basu, A., Lin, A., and Ramanathan, S. (2003). Routing using potentials: A dynamic traffic-aware routing algorithm. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communicatios (SIGCOMM)*, pages 37–48, Karlsruhe, Germany.

[Baumann et al., 2006] Baumann, R., Heimlicher, S., Strasser, M., and Weibel, A. (2006). A survey on routing metrics. TIK Report 262, ETH-Zentrum, Zurich, Switzerland.

[Beckmann et al., 1956] Beckmann, M., B., M. C., and B., W. C. (1956). *Studies in the Economics of Transportation.* Yale University Press, New Haven, USA.

[Bellman, 1957] Bellman, R. E. (1957). *Dynamic Programming.* Princeton University Press, Princeton, New Jersey, USA.

[Bertsekas, 1981] Bertsekas, D. P. (1981). Projected newton methods for optimization problems with simple contraints. In *Proceedings of the 20th IEEE Conference on Decision and Control Including the Symposium on Adaptive Processes (CDC)*, pages 762–767, San Diego, California, USA.

[Bertsekas, 1982] Bertsekas, D. P. (1982). Dynamic behavior of shortest path routing algorithm for communication networks. *IEEE Transactions on Automatic Control*, 27(1):60–74.

[Bertsekas and Gafni, 1983] Bertsekas, D. P. and Gafni, E. (1983). Projected newton methods and optimization of multicommodity flows. *IEEE Trans. on Automatic Control*, AC-28:139–159.

[Bertsekas et al., 1984] Bertsekas, D. P., Gafni, E. M., and Gallager, R. G. (1984). Second derivative algorithms for minimum delay distributed routing in networks. *IEEE Transactions on Communications*, 32(8):911–919.

[Bertsekas and Gallager, 1992] Bertsekas, D. P. and Gallager, R. G. (1992). *Data Networks (2nd edition).* Prentice Hall.

[Bianchi, 2000] Bianchi, G. (2000). Performance analysis of the ieee 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–547.

[Bickson, 2009] Bickson, D. (2009). *Gaussian Belief Propagation: Theory and Application.* Thesis for the degree of doctor of philosophy, The Hebrew University of Jerusalem.

[Bickson et al., 2009] Bickson, D., Tock, Y., Zymnis, A., Boyd, S., and Dolev, D. (2009). Distributed large scale network utility maximization. In *Proceedings of the 2009 IEEE International Symposium on Information Theory (ISIT)*, pages 829–833, Coex, Seoul, Korea.

[Bisnik and Abouzeid, 2009] Bisnik, N. and Abouzeid, A. A. (2009). Queuing network models for delay analysis of multihop wireless ad hoc networks. *Ad Hoc Networks*, 7(1):79 – 97.

[Borkar and Kumar, 2003] Borkar, V. S. and Kumar, P. R. (2003). Dynamic cesaro-wardrop equilibration in networks. *IEEE Transactions on Automatic Control*, 48(3):382–396.

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[Broder and Mitzenmacher, 2004] Broder, A. and Mitzenmacher, M. (2004). Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509.

[Busch et al., 2003] Busch, C., Surapaneni, S., and Tirthapura, S. (2003). Analysis of link reversal routing algorithms for mobile ad hoc networks. In *Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 210–219, Barcelona, Spain.

[Cantor and Gerla, 1974] Cantor, D. G. and Gerla, M. (1974). Optimal routing in a packet-switched computer network. *IEEE Transactions on Computers*, c-23(10):1062 – 1069.

[Cassandras et al., 1990] Cassandras, C., Abidi, M., and Towsley, D. (1990). Distributed routing with on-line marginal delay estimation. *IEEE Transactions on Communication*, 18:348 - 359.

[Chen et al., 2006] Chen, L., Low, S. H., Chiang, M., and Doyle, J. C. (2006). Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks. In *Proceedings of the 25th IEEE International Conference on Computer Communications*, pages 1–13, Barcelona, Catalunya, Spain.

[Chiang et al., 2007] Chiang, M., Low, S. H., Calderbank, A. R., and Doyle, J. C. (2007). Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312.

[Clausen and Jacquet, 2003] Clausen, T. and Jacquet, P. (2003). Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental).

[Cormen et al., 2009] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms, Third Edition.* MIT Press.

[Couto et al., 2003] Couto, D. S. J. D., Aguayo, D., Bicket, J., and Morris, R. (2003). A high-throughput path metric for multi-hop wireless routing. In *Proceedings of 9th ACM Annual International Conference on Mobile Computing and Networking (MobiCom '03)*, pages 134–146, San Diego, CA, USA.

[Dafermos and Sparrow, 1969] Dafermos, S. C. and Sparrow, F. T. (1969). The traffic assignment problem for a general network. *Journal of Research of the National Bureau of Standards*, 73B:91-118.

[Dai et al., 2008] Dai, L., Xue, Y., Chang, B., Cao, Y., and Cui, Y. (2008). Optimal routing for wireless mesh networks with dynamic traffic demand. *Mobile Networks and Applications*, 13(1-2):97–116.

[Dantzig, 1963] Dantzig, G. B. (1963). *Linear Programming and Extensions.* Princeton University Press, Princeton, New Jersey, USA.

[Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

[Dijkstra and Scholten, 1980] Dijkstra, E. W. and Scholten, C. S. (1980). Termination detection for diffusing computations. *Information Processing Letters*, 11(1):1–4.

[Dowling et al., 2005] Dowling, J., Curran, E., Cunningham, R., and Cahill, V. (2005). Using feedback in collaborative reinforcement learning to adaptively optimize manet routing. *IEEE Transactions on Systems, Man and Cybernetics*, 35(1):360-372.

[Draves et al., 2004] Draves, R., Padhye, J., and Zill, B. (2004). Routing in multi-radio, multi-hop wireless mesh networks. In *Proceedings of 10th ACM Annual International Conference on Mobile Computing and Networking (MobiCom '04)*, pages 114–128, Philadelphia, PA, USA.

[Facchinei et al., 1998] Facchinei, F., Judice, J., and Soares, J. (1998). An active set newton algorithm for large-scale nonlinear programs with box constraints. *SIAM Journal on Optimization*, 8(1):158–186.

[Fischer et al., 2006] Fischer, S., Kammenhuber, N., and Feldmann, A. (2006). Replex: dynamic traffic engineering based on wardrop routing policies. In *Proceedings of the 2006 ACM CoNEXT conference*, pages 1-12, New York, NY, USA.

[Floyd and Paxson, 2001] Floyd, S. and Paxson, V. (2001). Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking*, 9(4):392 – 403.

[Frank and Wolfe, 1956] Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110.

[Gafni and Bertsekas, 1981] Gafni, E. M. and Bertsekas, D. P. (1981). Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Transactions on Communications*, 29(1):11–18.

[Gallager, 1977] Gallager, R. G. (1977). A minimum delay routing algorithm using distributed computation. *IEEE Transactions on Communications*, 25(1):73–85.

[Garcia-Luna-Aceves et al., 2003] Garcia-Luna-Aceves, J. J., Mosko, M., and Perkins, C. E. (2003). A new approach to on-demand loop-free routing in ad hoc networks. In *Proceedings of the 23nd annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 53 – 62, New York, NY, USA.

[Garcia-Luna-Aceves and Murthy, 1997] Garcia-Luna-Aceves, J. J. and Murthy, S. (1997). A path-finding algorithm for loop-free routing. *IEEE/ACM Transactions on Networking*, 5(1):148–160.

[Garcia-Luna-Aceves and Roy, 2005] Garcia-Luna-Aceves, J. J. and Roy, S. (2005). On-demand loop-free routing with link vectors. *IEEE Journal of Selected Areas in Communications*, 23(3):533–546.

[Garcia-Lunes-Aceves, 1993] Garcia-Lunes-Aceves, J. J. (1993). Loop-free routing using diffusing computations. *IEEE/ACM Transactions on Networking*, 1(1):130–141.

[Gross et al., 2008] Gross, D., Shortle, J. F., Thompson, J. M., and Harris, C. M. (2008). *Fundamentals of Queueing Theory (4th Edition)*. John Wiley and Sons, Inc.

[Gupta and Shroff, 2011] Gupta, G. R. and Shroff, N. B. (2011). Delay analysis and optimality of scheduling policies for multi-hop wireless networks. *IEEE/ACM Transactions on Networking*, 19(1):129–141.

[Gupta and Kumar, 1997] Gupta, P. and Kumar, P. R. (1997). A system and traffic dependent adaptive routing algorithm for ad hoc networks. In *Proceedings of the 36th IEEE Conference on Decision and Control (CDC)*, volume 3, pages 2375–2380, San Diego, CA, USA.

[Gupta and Kumar, 2000] Gupta, P. and Kumar, P. R. (2000). The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404.

[Guven et al., 2004] Guven, T., Kommareddy, C., La, R. J., Shayman, M. A., and Bhattacharjee, B. (2004). Measurement based optimal multi-path routing. In *Proceedings*

of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), pages 187–196, Hong Kong, China.

[Hager and Zhang, 2006] Hager, W. W. and Zhang, H. (2006). A new active set algorithm for box constrained optimization. *SIAM Journal on Optimization*, 17(2):526–557.

[He et al., 2007] He, J., Bresler, M., Chiang, M., and Rexford, J. (2007). Towards robust multi-layer traffic engineering: Optimization of congestion control and routing. *IEEE Journal on Selected Areas in Communications*, 25(5):868 – 880.

[Humblet, 1991] Humblet, P. A. (1991). Another adaptive distributed shortest path algorithm. *IEEE Transactions on Communications*, 39(6):995–1003.

[Jadbabaie et al., 2009] Jadbabaie, A., Ozdaglar, A. E., and Zargham, M. (2009). A distributed newton method for network optimization. In *Proceedings of the 48th IEEE Conference on Decision and Control (CDC)*, pages 2736-2741, Shanghai, China.

[Jagabathula and Shah, 2008] Jagabathula, S. and Shah, D. (2008). Optimal delay scheduling in networks with arbitrary constraints. In *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 395–406, Annapolis, Maryland, USA.

[Jain et al., 2003] Jain, K., Padhye, J., Padmanabhan, V. N., and Qiu, L. (2003). Impact of interference on multi-hop wireless network performance. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, pages 66–80, San Diego, California, USA.

[Johnson et al., 2007] Johnson, D., Hu, Y., and Maltz, D. (2007). The dynamic source routing protocol (dsr) for mobile ad hoc networks for ipv4. RFC 4728 (Experimental).

[Johnson et al., 2009] Johnson, J. K., Bickson, D., and Dolev, D. (2009). Fixing convergence of gaussian belief propagation. In *Proceedings of the 2009 International*

Symposium on Information Theory (ISIT), volume 3, pages 1674 – 1678, Coex, Seoul, Korea.

[Kandula et al., 2005] Kandula, S., Katabi, D., Davie, B., and Charny, A. (2005). Walking the tightrope: Responsive yet stable traffic engineering. In *Proceedings of the 2005 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 253–264, Philadelphia, PA, USA.

[Kelly et al., 1998] Kelly, F. P., Maulloo, A. K., and Tan, D. (1998). Rate control in communication networks: Shadow price, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237 – 252.

[Kleinrock and Levy, 1988] Kleinrock, L. and Levy, H. (1988). The analysis of random polling systems. *INFORMS Operations Research*, 36(5):716–732.

[Kolar and Abu-Ghazaleh, 2006] Kolar, V. and Abu-Ghazaleh, N. B. (2006). A multi-commodity flow approach for globally aware routing in multi-hop wireless networks. In *Proceedings of the 4th Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM '06)*, pages 308–317, Pisa Italy.

[Kolar et al., 2009] Kolar, V., Abu-Ghazaleh, N. B., and Mahonen, P. (2009). Decomposition for low-complexity near-optimal routing in multi-hop wireless networks. In *Proceedings of the 2009 International Conference on Communications (ICC)*, pages 5302–5307, Dresden, Germany.

[Kulkarni and Glazebrook, 2002] Kulkarni, V. G. and Glazebrook, K. D. (2002). Output analysis of a single-buffer multiclass queue: Fcfs service. *Journal of Applied Probability*, 39(2):341 – 358.

[Kvalbein et al., 2009] Kvalbein, A., Dovrolis, C., and Muthu, C. (2009). Multipath load-adaptive routing: Putting the emphasis on robustness and simplicity. In *Pro-*

ceedings of the 17th IEEE International Conference on Network Protocols (ICNP), pages 203–212, Princeton, New Jersey, USA.

[L. R. Ford and Fulkerson, 1962] L. R. Ford, J. and Fulkerson, D. R. (1962). *Flows in Networks*. Princeton University Press, Princeton, New Jersey, USA.

[Lee and Gerla, 2001] Lee, S.-J. and Gerla, M. (2001). Split multipath routing with maximally disjoint paths in ad hoc networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 3201–3205, Helsinki, Finland.

[Lee, 1997] Lee, T. Y. S. (1997). A closed form solution for the asymmetric random polling system with correlated levy input process. *INFORMS Mathematics of Operations Research*, 22(2):432–457.

[Levchenko et al., 2008] Levchenko, K., Voelker, G. M., Paturi, R., and Savage, S. (2008). Xl: An efficient netork routing algorithm. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, pages 15–26, Seattle, WA, USA. ACM.

[Li et al., 2008] Li, B., Zhang, Q., Zhao, B., Yang, B., and Guan, X. (2008). The end-to-end rate control in multiple-hop wireless networks: Cross-layer formulation and optimal allocation. *IEEE Journal on Selected Areas in Communications*, 26(4):719–731.

[Lin and Shroff, 2006] Lin, X. and Shroff, N. B. (2006). Utility maximization for communication networks with multipath routing. *IEEE Transactions on Automatic Control*, 51(5):766 – 781.

[Littman and Boyan, 1993] Littman, M. and Boyan, J. (1993). A distributed reinforcement learning scheme for network routing. In *Proceedings of International Workshop on Applications of Neural Networks to Telecommunications*, pages 935–942.

[Liu and Sherali, 2012] Liu, J. and Sherali, H. D. (2012). A distributed newton approach for joint multi-hop routing and flow control: Theory and algorithm. In *Proceedings of the 31st IEEE International Conference on Computer and Communications (INFOCOM)*, pages 2489 – 2497, Orlando, Florida USA.

[Low and Lapsley, 1999] Low, S. H. and Lapsley, D. E. (1999). Optimization flow control-i: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874.

[Malkin, 1998] Malkin, G. (1998). Rip version 2. RFC 2453 (Standard). Updated by RFC 4822.

[Malone et al., 2007] Malone, D., Duffy, K., and Leith, D. (2007). Modeling the 802.11 distributed coordination function in non-saturated heterogeneous conditions. In *IEEE/ACM Transaction on Networking*, volume 15, Issue 1, pages 159 – 172.

[Marina and Das, 2002] Marina, M. K. and Das, S. R. (2002). Ad hoc on-demand multipath distance vector routing. *SIGMOBILE Mobile Computing and Communications Review*, 6:92–93.

[MATLAB, 2012] MATLAB (2012). *version 8.0.0.783 (R2012b)*. The MathWorks Inc., Natick, Massachusetts.

[Mosk-Aoyama et al., 2010] Mosk-Aoyama, D., Roughgarden, T., and Shah, D. (2010). Fully distributed algorithms for convex optimization problems. *SIAM Journal on Optimization*, 20(6):3260–3279.

[Moy, 1998] Moy, J. (1998). Ospf version 2. RFC 2328 (Standard). Updated by RFCs 5709, 6549.

[Mueller and Ghosal, 2005] Mueller, S. and Ghosal, D. (2005). Analysis of a distributed algorithm to determine multiple routes with path diversity in ad hoc networks. In

Proceedings of the 3rd International Symposium on Modeling and Optimization in Mobile Ad Hoc and Wireless Networks, pages 277–285, Garda, Trentino, Italy.

[Nedic and Ozdaglar, 2009] Nedic, A. and Ozdaglar, A. (2009). Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61.

[Nelakuditi and Zhang, 2002] Nelakuditi, S. and Zhang, Z.-L. (2002). A localized adaptive proportioning approach to qos routing. *IEEE Communications Magazine*, 40(6):66–71.

[Nie et al., 2004] Nie, Y., Zhang, H., and Lee, D.-H. (2004). Models and algorithms for the traffic assignment problem with link capacity constraints. *Elsevier Transportation Research Part B*, 38(4):285–312.

[Oran, 1990] Oran, D. (1990). Osi is-is intra-domain routing protocol. RFC 1142 (Informational).

[Palomar and Chiang, 2006] Palomar, D. P. and Chiang, M. (2006). A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications*, 24(8):1439–1451.

[Papadimitratos et al., 2002] Papadimitratos, P., Haas, Z. J., and Sirer, E. G. (2002). Path set selection in mobile ad hoc networks. In *Proceedings of the 3rd Internation Symposium on Mobile Ad Hoc Networking and Computing*, pages 1 – 11, New York, NY, USA. ACM.

[Park and Corson, 1997] Park, V. D. and Corson, M. S. (1997). A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1405–1413, Kobe, Japan.

[Paxson and Floyd, 1995] Paxson, V. and Floyd, S. (1995). Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226 – 244.

[Pei et al., 2000] Pei, G., Gerla, M., and Chen, T.-W. (2000). Fisheye state routing in mobile ad hoc networks. In *Proceedings of the 2000 ICDCS Workshops*, pages 71–78, Taipei, Taiwan.

[Perkins et al., 2003] Perkins, C., Belding-Royer, E., and Das, S. (2003). Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental).

[Perkins et al., 2012] Perkins, C., Chakeres, I., and CenGen (2012). Dynamic MANET On-Demand (AODVv2) Routing. IETF Internet-Draft.

[Perkins and Bhagwat, 1994] Perkins, C. E. and Bhagwat, P. (1994). Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *Proceedings of the ACM SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, London, UK.

[Purkayastha and Baras, 2008] Purkayastha, P. and Baras, J. S. (2008). An optimal distributed routing algorithm using dual decomposition techniques. *Communications in Information and Systems*, 8(3):277–302.

[Raghunathan and Kumar, 2009] Raghunathan, V. and Kumar, P. R. (2009). Wardrop routing in wireless networks. *IEEE Transaction on Mobile Computing*, 8(5):636–652.

[Roughgarden, 2002] Roughgarden, T. (2002). *Selfish Routing.* PhD thesis, Cornell University, Ithaca, NY 14850.

[Roughgarden and Tardos, 2002] Roughgarden, T. and Tardos, E. (2002). How bad is selfish routing? *Journal of the ACM*, 49(2):236-259.

[Roy and Garcia-Luna-Aceves, 2001] Roy, S. and Garcia-Luna-Aceves, J. J. (2001). Using minimal source trees for on-demand routing in ad hoc networks. In *Proceedings*

of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Anchorage, Alaska, USA.

[Roy et al., 2003] Roy, S., Saha, D., Bandyopadhyay, S., Tanaka, S., and Ueda, T. (2003). Improving end-to-end delay through load balancing with multipath routing in ad hoc wireless networks using directional antenna. In *Proceedings of the 7th International Workshop on Distributed Computing (IWDC)*, pages 225–234, Kolkata, India.

[Sadiq et al., 2011] Sadiq, B., Baek, S. J., and de Veciana, G. (2011). Delay-optimal opportunistic scheduling and approximations: The log rule. *IEEE/ACM Transactions on Networking*, 19(2):405–418.

[Saltzer et al., 1984] Saltzer, J. H., Reed, D. P., and Clark, D. D. (1984). End-to-end arguments in system design. *ACM Transactions on Computer Systems (TOCS)*, 2(4):277 – 288.

[Segall and Sidi, 1981] Segall, A. and Sidi, M. (1981). A failsafe distributed protocol for minimum delay routing. *IEEE Transactions on Communications*, 29(5):689–695.

[Sharma et al., 2007] Sharma, G., Mazumdar, R., and Shroff, N. B. (2007). Delay and capacity trade-offs in mobile ad hoc networks: A global perspective. *IEEE/ACM Transactions on Networking*, 15(5):981–992.

[Shenker, 1995] Shenker, S. (1995). Fundamental design issues for the future internet. *IEEE Journal on Selected Areas in Communications*, 13(7):1176 – 1188.

[Shental et al., 2008] Shental, O., Paul H, S., Wolf, J. K., Bickson, D., and Dolev, D. (2008). Gaussian belief propagation solver for systems of linear equations. In *Proceedings of the 2008 IEEE International Symposium on Information Theory (ISIT)*, pages 1863–1867, Toronto, Ont., Canada.

[Shewchuk, 1994] Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University, Pittsburgh, PA, USA.

[Silva et al., 2008] Silva, A., Altman, E., and Bernhard, P. (2008). Numerical solutions of continuum equilibria for routing in dense ad-hoc networks. In *Proceedings of the 3rd International Conference on Performance Evaluation Methodologies and Tools*, pages 1-8, Brussels, Belgium. ICST.

[Sobrinho, 2003] Sobrinho, J. L. (2003). Network routing with path vector protocols: Theory and applications. In *Proceedings of the 2003 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communicatios (SIGCOMM)*, pages 49–60, Karlsruhe, Germany.

[Tan et al., 2006] Tan, C. W., Palomar, D. P., and Chiang, M. (2006). Distributed optimization of coupled systems with applications to network utility maximization. In *Proceedings of the 2006 IEEE Internation Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 5, pages 981–984, Toulouse, France.

[Tian and Cheng, 2012] Tian, X. and Cheng, Y. (2012). Loop mitigation in bloom filter based multicast: A destination-oriented approach. In *Proceedings of the 31st IEEE International Conference on Computer and Communications (INFOCOM)*, pages 2131–2139, Orlando, Florida USA.

[Tickoo and Sikdar, 2008] Tickoo, O. and Sikdar, B. (2008). Modeling queueing and channel access delay in unsaturated ieee 802.11 random access mac based wireless networks. *IEEE/ACM Transactions on Networking*, 16(4):878–891.

[Toumpis, 2006] Toumpis, S. (2006). Optimal design and operation of massively dense wireless networks: or how to solve 21st century problems using 19th century mathematics. In *Proceedings of the 2006 ACM workshop on Interdisciplinary systems ap-*

proach in performance evaluation and design of computer & communications sytems, page 7, New York, NY, USA.

[Vutukury and Garcia-Luna-Aceves, 1999] Vutukury, S. and Garcia-Luna-Aceves, J. J. (1999). A simple approximation to minimum-delay routing. In *Proceedings of the ACM SIGCOMM Conference on Applications, technologies, architectures, and protocols for computer communication*, pages 227-238, New York, USA.

[Vutukury and Garcia-Luna-Aceves, 2000a] Vutukury, S. and Garcia-Luna-Aceves, J. J. (2000a). Mpath: A loop-free multipath routing algorithm. *Elsevier Journal of Microprocessors and Microsystems*, 24(6):319–327.

[Vutukury and Garcia-Luna-Aceves, 2000b] Vutukury, S. and Garcia-Luna-Aceves, J. J. (2000b). A traffic engineering approach based on minimum-delay routing. In *Proceedings of 9th IEEE International Conference on Computer Communications and Networks (ICCCN)*, pages 42 –47.

[Vutukury and Garcia-Luna-Aceves, 2001] Vutukury, S. and Garcia-Luna-Aceves, J. J. (2001). Mdva: A distance-vector multipath routing protocol. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 1, pages 557–564, Anchorage, Alaska, USA.

[Wang et al., 2003] Wang, W.-H., Palaniswami, M., and Low, S. H. (2003). Optimal flow control and routing in multi-path networks. *Elsevier Performance Evaluation*, 52(2-3):119–132.

[Wang and Crowcroft, 1992] Wang, Z. and Crowcroft, J. (1992). Analysis of shortest-path routing algorithms in a dynamic network environment. *ACM Computer Communication Review*, 22(2):63–71.

[Wardrop, 1952] Wardrop, J. G. (1952). Some theoretical aspects of road traffic research. In *Proceedings of Institute of Civil Engineers.*, volume 1, pages 325–378.

[Wei et al., 2010] Wei, E., Ozdaglar, A. E., and Jadbabaie, A. (2010). A distributed newton method for network utility maximization. In *Proceedings of the 49th IEEE Conference on Decision and Control, CDC*, pages 1810 – 1815, Atlanta, Georgia, USA.

[Wei et al., 2011] Wei, E., Zargham, M., Ozdaglar, A. E., and Jadbabaie, A. (2011). On dual convergence of the distributed newton method for network utility maximization. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 6612–6617, Orlando, Florida USA.

[Xi and Yeh, 2006] Xi, Y. and Yeh, E. M. (2006). Node-based distributed optimal control of wireless networks. In *Proceedings of the 40th Annual IEEE Conference on Information Science and Systems (CISS)*, pages 1566–1571, Princeton, New Jersey, USA.

[Xiao et al., 2004] Xiao, L., Johansson, M., and Boyd, S. (2004). Simulataneous routing and resource allocation via dual decomposition. *IEEE Transactions on Communications*, 52(7).

[Xu et al., 2007] Xu, D., Chiang, M., and Rexford, J. (2007). Deft: Distributed exponentially-weighted flow splitting. In *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM)*, pages 71–79, Anchorage, Alaska, USA.

[Xu et al., 2011] Xu, D., Chiang, M., and Rexford, J. (2011). Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. *IEEE/ACM Transactions on Networking*, 19(6):1717 – 1730.

[Yang and Weber, 2011] Yang, G. and Weber, S. (2011). Adapting wardrop equilibrium to facilitate optimal routing in wireless ad hoc networks. In *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1 – 9, Lucca, Italy.

[Yang et al., 2005] Yang, Y., Wang, J., and Kravets, R. (2005). Designing routing metrics for mesh networks. In *Proceedings of the 1st IEEE Workshop on Wireless Mesh Networks (WiMesh)*, Santa Clara, CA, USA.

[Ye et al., 2003] Ye, Z., Krishnamurthy, S. V., and Tripathi, S. K. (2003). A framework for reliable routing in mobile ad hoc networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications*, pages 270–280, San Francisco, USA.

[Yi and Shakkottai, 2007] Yi, Y. and Shakkottai, S. (2007). Hop-by-hop congestion control over a wireless multi-hop network. *IEEE/ACM Transactions on Networking*, 15(1):133–144.

[Zargham et al., 2012] Zargham, M., Ribeiro, A., and Jadbabaie, A. (2012). A distributed line search for network optimization. In *Proceedings of the 2012 American Control Conference (ACC)*, Montreal, Canada.

[Zaumen and Luna-Aceves, 1998] Zaumen, W. T. and Luna-Aceves, J. J. G. (1998). Loop-free multipath routing using generalized diffusing computations. In *Proceedings of the 17th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 3, pages 1408 – 1417. IEEE.

[Zhang et al., 2005] Zhang, C., Liu, Y., Gong, W., Kurose, J., Moll, R., and Towsley, D. (2005). On optimal routing with multiple traffic matrices. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 607–618, Miami, Florida, USA.

[Zymnis et al., 2007] Zymnis, A., Trichakis, N., Boyd, S., and O'Neill, D. (2007). An interior-point method for large scale network utility maximization. In *Proceedings of the 45th SIAM Annual Allerton Conference on Communication, Control, and Computing*, pages 877–881, Monticello, Illinois, USA.