UNIVERSITY OF CALIFORNIA SAN DIEGO

Destination-based Routing and Circuit Allocation for Future Traffic Growth

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Computer Engineering)

by

Ping Yin

2020

DEDICATION

To my family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

The pursuit of my Ph.D degree at UCSD is incredibly challenging in the past few years. However, I have been very fortunate to have so many intelligent people help me during the long journey, without whom it would have been impossible for me to achieve my Ph.D.

First and foremost, I wish to express my sincere appreciation to my advisor, Professor Bill Lin, for his support during the last few years, both academically and financially. I have very much appreciated his insightful comments on our projects, which have turned out to be very useful and helped me make progress on the right track. I have also appreciated his efforts in finding resources to help me overcome academic weakness. Throughout these years, his patient and enlightening guidance has proved to be invaluable.

I would like to thank Professor Stephen Boyd from Stanford University, Professor Jim Dai from Cornell University, Professor Jun Xu from Georgia Tech, and my friend, Dr. Sen Yang, for their generous help in our collaboration on my Ph.D research.

I would also like to thank my doctoral committee members for their reviews and advises of this thesis.

Finally, I wish to acknowledge the support and great love of my family, my father, Chengxin Yin; my mother, Quanying Li; and my beloved grandmother, Zuxiu Peng. They have been very supportive on my education and life.

Chapter 2, in part, is a reprint of the material as it appears in ACM Transactions on Modeling and Performance Evaluation of Computing Systems, January 2019. Yin, Ping; Yang, Sen; Xu, Jun; Dai, Jim; Lin, Bill. Chapter 3 also includes the materials as it appears in ACM/IEEE Symposium on Architectures for Networking and Communications Systems Conference, 2017. Yin, Ping; Yang, Sen; Xu, Jun; Dai, Jim; Lin, Bill. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in full, is a reprint of the materials as it appears in Optimization and

Engineering, 2019. Yin, Ping; Diamond, Steven; Lin, Bill; Boyd, Stephen. The dissertation author was the primary investigator and author of these papers.

Chapter 4, in full, is a reprint of the material as it appears in IEEE 39th International Conference on Distributed Computing Systems, 2019. Zhu, Yuqing; Yin, Ping; Li, Deying; Lin, Bill. The dissertation author was the primary investigator and author of this paper.

VITA

2013	Bachelor of Science in Electrical Engineering Shanghai Jiao Tong University
2015	Master of Science in Electrical Engineering (Computer Engineering) University of California San Diego
2020	Doctor of Philosophy in Electrical Engineering (Computer Engineering) University of California San Diego

PUBLICATIONS

Ping Yin, Sen Yang, Jun Xu, Jim Dai, Bill Lin "Improving Backpressure-based Adaptive Routing via Incremental Expansion of Routing Choices," ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), (2017)

Ping Yin, Sen Yang, Jun Xu, Jim Dai, Bill Lin "Efficient Traffic Load-Balancing via Incremental Expansion of Routing Choices," ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS), January 2019 Article No.: 1

Ping Yin, Steven Diamond, Bill Lin, Stephen Boyd "Network Optimization for Unified Packet and Circuit Switched Networks," Optimization and Engineering, (2019).

Yuqing Zhu, Ping Yin, Deying Li, Bill Lin "Strengthening the Positive Effect of Viral Marketing," IEEE 39th International Conference on Distributed Computing Systems (ICDCS), (2019) ABSTRACT OF THE DISSERTATION

Destination-based Routing and Circuit Allocation for Future Traffic Growth

by

Ping Yin

Doctor of Philosophy in Electrical Engineering (Computer Engineering)

University of California San Diego, 2020

Professor Bill Lin, Chair

Internet traffic continues to grow relentlessly, driven largely by increasingly high-resolution video streaming, the increasing adoption of cloud computing, the emergence of 5G networks, and the ever-growing reach of social media and social networks. Existing networks use packet switching to route packets on a hop-by-hop basis from the source to the destination. However, they suffer from two shortcomings. First, in existing networks, packets are routed along a fixed shortest path using the Open Shortest Path First (OSPF) protocol or obliviously load-balanced across equal-cost paths using the Equal-Cost Multi-Path (ECMP) protocol. These routing protocols do not fully utilize the network capacity because they do not adapt to network congestions in their routing decisions. Second, although studies have shown that the majority of packets processed by Internet routers are pass-through traffic, packets nonetheless have to be queued and routed at every hop in existing networks, which unnecessarily adds substantial delays and processing costs.

In this thesis, we present two new approaches to overcome these shortcomings. First, we propose new backpressure-based routing algorithms which use only shortest-path routes when they are sufficient to accommodate the given traffic load, but will incrementally expand routing choices as needed to accommodate increasing traffic loads. This avoids the poor delay performance inherent in backpressure-based routing algorithms where packets may take long detours under light or moderate loads, and still retains the notable advantage, the network-wide optimal throughput, because packets are adaptively routed along less congested paths.

Second, we propose a unified packet and circuit switched network in which the underlying optical transport is used to circuit-switch pass-through traffic by means of pre-established circuits. This avoids unnecessary packet queuing delays and processing costs at each hop. We propose a novel convex optimization framework based on a new destination-based multicommodity flow formulation for the allocation of circuits in such unified networks.

Finally, social networks have become extremely large and complicated. We present an approach to an influence spreading optimization problem that maximizes the positive effects in a large social network when negative people who are inclined to evaluate a product negatively are present.

Chapter 1

Introduction

1.1 Motivation

The Internet is experiencing explosive traffic growth. As an observation expressed as Edholm's law, Internet bandwidth has been doubling every 18 months. Driven largely by increasingly high-resolution video streaming, the increasing adoption of cloud computing, the emergence of 5G networks, and the evergrowing reach of social media and social networks, Internet traffic will continue to grow relentlessly. Existing networks use packet switching to route packets on a hop-by-hop basis from the source to the destination. However, they suffer from two shortcomings.

First, in existing networks, packets are routed along a fixed shortest path using the Open Shortest Path First (OSPF) protocol or obliviously load-balanced across equal-cost paths using the Equal-Cost Multi-Path (ECMP) protocol. These routing protocols do not fully utilize the network capacity because they do not adapt to network congestions in their routing decisions. Compared with these fixed routing protocols, where each traffic flow has a single and fixed route, adaptive routing is more appealing because it tends to offer better throughput by routing packets along different routing paths depending on network congestion. Among adaptive routing algorithms, backpressure-based algorithms [78, 17, 86, 62, 69, 6, 70, 5, 9, 87] have been extensively studied in the literature because they have been shown to be network-wide throughput optimal. However, backpressure-based algorithms typically have poor delay performance under light or moderate loads because they allow the routing of packets to any adjacent node as the next-hop, even if the routing decisions will cause a packet to take unnecessarily long detours, potentially traversing routing loops. In addition, backpressure-based algorithms require to compute differential backlogs for every per-destination queue with the corresponding per-destination queue at every adjacent node. This computation cost is expensive given the large number of possible pairwise differential backlogs. Therefore, a new routing algorithm which offers competitively low delay and much less cost in computation while retaining the merit of network-wide optimal throughput is desirable.

Second, although studies have shown that the majority of packets processed by Internet routers are pass-through traffic, in existing networks, packets nonetheless have to be queued and routed at every hop in existing networks, which unnecessarily adds substantial delays and processing costs. These pass-through traffic packets can be better circuit-switched in a unified packet and circuit switched network through the underlying optical transport network by means of pre-established circuits to avoid substantial delays and processing costs associated with Internet routers. More recently, given the broad success of software-defined networking (SDN) [72, 49, 83, 55], there has been considerable renewed interest in the unified packet and circuit switched network architectures based on SDN as the unified control plane [30, 31]. In the SDN-based unified architecture proposed in [31], backbone routers are replaced with less expensive hybrid optical-circuit/electrical-packet switches that have both circuit-switching and packet-switching capabilities. These hybrid switches are logically connected in a fully-meshed network where each hybrid switch implements an IP node, and where each IP node is logically connected to each and every other IP node via a single direct circuit-switched hop. This unified packet and circuit-switched network can then be managed using a single converged control plane. A key problem that must be solved in this unified architecture approach is the allocation of optical circuits between adjacent IP nodes in the logical full-mesh, which is known as multi-commodity flow problems. Classically, multi-commodity flow formulations use $n(n-1)m$ flow assignment variables for a network consisting of n nodes and m edges, each of which defines the fraction of the corresponding IE pair traffic (among $n(n-1)$ IE pairs) along the corresponding edge (among m edges). This is obviously not scalable when n becomes too large, and its solution does not necessarily deliver fair sharing of network capacity among competing traffic flows. Therefore, a new compact formulation framework which is more scalable as n becomes large and also ensures fair sharing of network capacity among competing traffic flows is attractive.

1.2 Contributions and Organization

The contributions of this dissertation include new backpressure-based routing algorithms for packet switching networks which achieve low packet delay, optimal network-wide throughput, and small computation cost simultaneously; a novel convex optimization framework based on a new destination-based multi-commodity flow formulation for the allocation of circuits in unified packet and circuit switched networks; a new approach to an influence spreading optimization problem which maximizes the positive effect in a large social network.

First, we propose new backpressure-based routing algorithms that minimizes congestion by adaptively routing packets along less congested paths. To overcome the poor delay performance under light or moderate loads, we propose new backpressure-based adaptive routing algorithms that only use shortest-path routes to destinations when they are sufficient to accommodate the given traffic load, but the proposed algorithms will incrementally expand routing choices as needed to accommodate increasing traffic loads. In particular, we propose two route-expanding BP (backpressure) algorithms, called L-BP and A-BP, that are based on the incremental expansion of backpressure routing choices in response to congestion at a node on a per-destination queue basis. L-BP detects congestion by monitoring per-destination queue lengths, whereas A-BP detects congestion by monitoring the waiting times of packets at the heads of per-destination queues. Networks running these two algorithms are purely backpressure-based, but some per-destination queues are in shortest-path mode while others are allowed to be forwarded to any neighbor node. We also propose two semi-oblivious BP (backpressure) algorithms, called O-BP and E-BP, that combine OSPF and ECMP oblivious routing, respectively, with backpressure routing. Networks running these algorithms first use OSPF or ECMP oblivious routing when the traffic load is light or moderate, but these algorithms incrementally switch nodes on a per-node basis to backpressure routing in response to congestion detected by monitoring output port queue lengths. All of these new backpressure-based routing algorithms achieve the design goal of low packet delay, small computation cost, and optimal network-wide throughput simultaneously.

In addition to new adaptive routing algorithms for packet switching networks, we propose a new unified packet and circuit switched network architecture in which the underlying optical transport is used to circuit-switch pass-through traffic by means of pre-established circuits. This avoids unnecessary packet queuing delays and processing costs at each hop. We propose a novel convex optimization framework based on a new destination-based multicommodity flow formulation for the allocation of circuits in such unified networks. This formulation reduces the number of flow assignment variables by a factor of $n-1$, and thus the method may be scaled to far larger networks, where n is the number of nodes in the network.

In particular, we consider two deployment settings for circuit allocation. In the first setting, we consider the case in which real-time traffic measurements are possible, and we can dynamically allocate circuits on a frequent basis in response to changing traffic. In the second setting, we consider the case in which we allocate circuits based on historical traffic patterns. The past traffic measurements may be used to predict the behavior of future traffic and precompute the circuit configuration offline accordingly because previous studies have shown that the aggregate traffic at the core of the network tends to be very smooth and that it follows strong diurnal patterns over repeated data sets [22, 67, 56]. For both two deplayment settings, we formulate global network optimization objectives as concave functions that capture the fair sharing of network capacity among competing traffic flows. The convexity of our problem formulations ensures globally optimal solutions. In the case of traffic fluctuations or unexpected traffic changes which lead to inadequate pre-computed capacities along direct circuits, we propose two adaptive re-routing algorithms to adaptively re-route the excess traffic over circuits with spare capacity. One is based on a variant of the well-known backpressure-based re-routing algorithm [85] that guarantees optimal re-routing; the other is a simple but powerful greedy re-routing algorithm that simply re-routes the excess traffic over the outgoing circuit with the most residual capacity. With the help of adaptive re-routing, we can increase network throughput without the need to create new circuits on-the-fly.

The last problem considered in this thesis is related to social networks. Social networks have become extremely large and complicated. A key optimization problem in social networks is the influence spreading optimization problem in which the goal is to maximize the positive effects of an influence campaign in a large social network. We consider in this thesis a version of the problem in which people who are inclined to provide negative comments or viewpoints are present. In particular, the problem we study is how to choose the initial adopters (seeds) in the network to maximize the actually benefit, i.e., the difference of the total positive response and the total negative response. We call our problem *Strengthening the Positive Effect (SPE)* since it is apparently our goal. We check the objective function of *SPE* and find it is non-monotone and non-submodular. What is more, we prove that *SPE* cannot be solved with any positive guarantee unless P=NP. Since generally solving *SPE* with some guarantee is almost impossible, we break our problem into some special cases. We first investigate the unweighted and undirected network, and propose an almost optimal algorithm for a special case, and an approximate algorithm for the general case. For general network, we give a reasonable constraint and devise an approximation algorithm.

The rest of the dissertation is organized as follows:

- Chapter 2 presents a series of new backpressure-based routing algorithms by means of incremental expansion of routing choices as needed for packet switching network.
- Chapter 3 presents a novel convex optimization framework based on a new destination-based multicommodity flow formulation for the allocation of circuits in unified packet and circuit switched networks.
- Chapter 4 presents a new approach to the influence spreading optimization problem for large social networks that considers the negative impact of negative users.
- Chapter 5 concludes this dissertation.

Chapter 2

Efficient Traffic Load-Balancing via Incremental Expansion of Routing Choices

2.1 Introduction

Traffic on the Internet continues to grow at a rapid pace. The choice of routing algorithm plays a vital role in the performance of communication networks. Compared with fixed routing [17, 43, 42], where each flow has a single and fixed route, adaptive routing is more appealing because it offers better latency and throughput by routing packets along different routing paths, depending on network congestion. Among adaptive routing algorithms, backpressure-based algorithms [78, 17, 86, 62, 69, 6, 70, 5, 9, 87] have been extensively studied in the literature because they have been shown to be network-wide throughput optimal [78]. It was initially introduced in the context of wireless radio networks, but it can be easily adapted to wireline networks as well, for example for packet routing in backbone networks for the Internet. Despite throughput-optimality guarantees, backpressure-based algorithms have not been used in practice due to several shortcomings.

First, backpressure-based algorithms typically have poor delay performance under light or moderate loads because packets may be sent over unnecessarily long routes, potentially traversing routing loops. The original backpressure algorithm allows the routing of a packet to any adjacent node as the next-hop, even if the routing decisions will cause a packet to take long detours. Second, backpressure algorithms typically maintain per-destination queues, and the routing and scheduling decisions are based on maintaining differential backlogs for every per-destination queue with the corresponding per-destination queue at every adjacent node. Although the implementation of per-destination queues has often been cited as a concern [17, 43, 42, 86], we note that significant advances have been made in memory architectures since the original backpressure routing work for implementing huge packet buffers at line rates that support a very large number of logical queues [41, 73, 81, 80][1].

Despite these advances that address the practical implementation of per-destination queues, the need remains for backpressure-based algorithms to compute differential backlogs for *every* per-destination queue with the corresponding per-destination queue at every adjacent node. This computation is expensive given the large number of possible pairwise differential backlogs. Further, the computation requires many exchanges of backlog information between every pair of adjacent nodes for every pair of per-destination queues. The substantial amount of computations and associated information exchanges remains significant impediments for practical implementations.

To address the poor delay performance concern, the backpressure idea can be applied to a fixed routing problem, where packets are forced to use shortest paths [17]. However, limiting routing choices shrinks the network stability region and is thus not throughput

[1]Some of these memory architectures have been in commercial use in modern Internet routers to support per-class queuing or per-flow queuing, where the number of logical queues far exceeds the number of destinations needed for backpressure routing.

optimal. As we shall later see in the evaluation section, limiting the routing choices to just shortest paths will cause the network to saturate much earlier than if all routing choices are permitted.

Several prior works [62, 61, 33, 86, 34] have recognized the importance of favoring shorter paths instead of only considering shortest paths. However, all these approaches still require the computation of differential backlogs for all per-destination queues between *every* pair of neighboring nodes and the associated exchanges of backlog information. In comparison, our proposed algorithms only need a small number of the computations and backlog information exchanges as needed. Although the approach studied in [86] further offers provably minimal-hop (MinHop) routing, their solution dramatically increases the number of queues that each node needs to maintain since their approach requires per-hop queues for each destination. The dramatic increase in the number of queues makes the associated differential backlog computation problem even more difficult. Besides, the delay performance using their MinHop approach depends on the choices of parameters used. As we shall see later in the evaluation section, our approach offers comparable delay performance with the carefully tuned MinHop algorithm, while significantly reducing the information exchange frequency.

Other algorithms have been proposed to reduce the delay without using shortest paths. [6, 5] have proposed to replicate packets adaptively to build up gradients towards the destinations faster. Not only does this approach need a duplicate buffer for each per-destination queue in a router, but managing the original packets with their replicas can be difficult in practice. Besides, the number of backlog information exchanges and computations between every pair of per-destination queues in their algorithm is the same as in the original backpressure algorithm. [25] proposed a class of enhanced BP algorithms by incorporating a general queue-dependent bias function, which basically use the queue state information beyond one hop, into the traditional backpressure algorithm. Although proven to be throughput optimal, their algorithms add the computation complexity to compute the bias function term in addition to what the original backpressure algorithm needs. Even their simplest BPnxt algorithm requires twice the amount of computations and backlog information exchanges as what the original backpressure algorithm requires. The other algorithm, BPmin, not only needs global queue state information, but also requires N times more computations than the original backpressure algorithm. Our approach achieves the same goal with a much simpler design, that is dynamically expanding the routing choices as needed in response to the network congestion.

Further, there are prior works [87, 43, 42] that aim to reduce the number of backlog information exchanges needed to compute the pairwise differential backlogs of per-destination queues. [87] proposed a cluster-based back-pressure algorithm, where nodes are grouped into clusters and each node needs only to maintain per-destination queues for destinations within its cluster. However, this approach, although retaining the same optimal throughput as the original backpressure algorithm, still has large delay at low or moderate traffic loads. The approaches in [43, 42] eliminate the per-flow information by using per-hop queues and per-link queues, however, it can only be applied to *fixed routing* scenarios, while our approach aims at *adaptive routing*, where the path that a packet traverses in the network is not pre-defined.

Finally, [17, 9] introduced the idea of shadow queues for making adaptive routing decisions. Their idea is to create a shadow network in which a backpressure algorithm is used to make routing decisions. Although their approach does not require per-destination queuing of packets, their solution still incurs the same calculation complexity as the original backpressure algorithm for the shadow queues in that the same computation of differential backlogs for all destinations between every pair of neighboring nodes and the same associated exchanges of backlog information are still required. Further, although their approach stores the actual packets in per-neighbor queues instead of per-destination queues, the amount of packet buffer storage that each node needs remains the same[2]. We believe that our solution is complementary to [17, 9] in that the algorithms described in this chapter can be used as the shadow algorithm in their framework.

Although these algorithms partly addressed the aforementioned disadvantages of the original backpressure algorithm, to the best of our knowledge, no work has addressed all the aforementioned issues in an *adaptive routing* scenario. Our approach substantially improves the delay performance, significantly reduces the number of pairwise differential backlog information exchanges and computations, but still retains the same optimal throughput as the original backpressure algorithm.

2.1.1 Our Approach

In this chapter, we propose two related classes of modified backpressure-based routing algorithms that address the aforementioned concerns. We first propose *route-expanding backpressure-based routing algorithms* that are based on the idea that routing choices should be limited to next-hops that are along shortest path routes by default. This approach significantly reduces the amount of differential backlog calculations and associated information exchanges as each node only has to consider a subset of next hops for each destination. In addition, this approach addresses delay performance concerns by only routing packets along shortest path routes when the traffic load is light or moderate.

In particular, we propose to detect congestion by monitoring per-destination queue lengths or the waiting times of packets in the per-destination queues. When the length of a per-destination queue or the waiting time of a packet at the head of a per-destination queue exceeds some threshold, the routing choices for the corresponding per-destination queue get *expanded* to include next hops that are not along shortest path routes. This

[2]The state-of-the-art DRAM-based packet buffers [41, 73, 81, 80] can store a huge number of packets, tens of gigabytes, and support a very large number of logical queues.

expansion of routing choices is on a per-destination queue basis. Although a packet may be forwarded to a next hop that is not along a shortest path route to the destination, the packet may still be forwarded along a shortest path route from this next hop to the destination if the corresponding per-destination queue at this next hop is not yet congested. This way, routing choices are *incrementally* expanded at different nodes in the network as needed with increasingly longer paths considered. In effect, a packet can take a detour whenever it encounters congestion along the way to the destination. When a node expands its routing choices for packets for a particular destination, it notifies other adjacent nodes to begin providing backlog information, and it expands its differential backlog calculations with those adjacent nodes as well.

Alternatively, we propose *semi-oblivious backpressure-based routing algorithms* that simply route packets *obliviously* by default, which only consider shortest-path next-hops. In this default *oblivious-routing mode*, backlog information exchanges and differential backlog calculations are largely avoided, and only shortest-path next-hops are considered, which also addresses delay performance concerns by only routing packets along shortest path routes when the traffic load is light or moderate. The proposed semi-oblivious approach detects congestion by monitoring output port queue lengths. When the length of an output queue at a router exceeds some threshold, the entire router switches to *backpressure-routing mode* where packets are *adaptively* routed to any adjacent node as possible next-hops using the backpressure routing algorithm. Although when a router is in backpressure-routing mode, a packet may be forwarded to a next hop that is not along a shortest path route to the destination, the packet may still be forwarded along a shortest path route from this next hop forward if the subsequent routers are still in oblivious-routing mode. This way, the proposed semi-oblivious approach also *incrementally* expands routing choices as needed with increasingly longer paths considered.

2.1.2 Contributions and Outline

The main contributions of this chapter are as follows:

- We propose two route-expanding BP (backpressure) algorithms, called L-BP and A-BP, that are based on the incremental expansion of backpressure routing choices in response to congestion at a node on a per-destination queue basis. L-BP detects congestion by monitoring per-destination queue lengths, whereas A-BP detects congestion by monitoring the waiting times of packets at the heads of per-destination queues. Networks running these two algorithms are purely backpressure-based, but some per-destination queues are in shortest-path mode while others are allowed to be forwarded to any neighbor node.

- We also propose two semi-oblivious BP (backpressure) algorithms, called O-BP and E-BP, that combine OSPF and ECMP oblivious routing, respectively, with backpressure routing. Networks running these algorithms first use OSPF or ECMP oblivious routing when the traffic load is light or moderate, but these algorithms incrementally switch nodes on a per-node basis to backpressure routing in response to congestion detected by monitoring output port queue lengths.

- We prove theoretically that all these algorithms are network-wide throughput optimal (i.e., the proposed algorithms can explore the same network stability region as the original backpressure algorithm) in A.1. In particular, we use a fluid model for our proofs, which models well the system dynamics of our modified algorithms.

- We extensively evaluate our proposed algorithms on the adaptive Internet routing problem. We show our evaluations on the Abilene network [40], a public PoP-level academic network in the US, using actual traffic profiles measured on the network. Our simulation results show that our proposed algorithms indeed provide substantial improvements in delay performance. Our simulation results further show that in practice, our approach dramatically reduces the number of pairwise differential backlogs that have to be computed and the amount of corresponding backlog information that has to be exchanged because routing choices are only incrementally expanded as needed. That is, only a subset of per-destination queues in a subset of nodes (in the case of route-expanding BP algorithms) or a subset of nodes (in the case of the semi-oblivious BP algorithm) need to consider expanded routing choices even for traffic loads that approach the edge of the network stability region.

- To the best of our knowledge, our approach is the first work which substantially improves the delay performance and also significantly reduces the number of pairwise differential backlog information exchanges and computations, while keeping the same optimal throughput benefit as the original backpressure algorithm in the adaptive routing scenarios.

The rest of the chapter is organized as follows: In Section 2.2, we present the basic network model and summarize the original backpressure algorithm. Then we present our route-expanding BP algorithms and semi-oblivious BP algorithms in Sections 2.3 and 2.4, respectively. In Section 2.5, we describe our experimental setup and simulation results. We conclude our chapter in Section 2.6.

2.2 Background

2.2.1 The Network Model

We consider a multi-hop network represented by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes, and $\mathcal{L}$ is the set of directed links. All packets that enter the network are associated with a particular *commodity* that corresponds to the packet *destination*. A packet that is destined for node c is regarded as a commodity c packet, $c = 1, \ldots, N$. We use $\mathcal{L}_c$ to denote the routing restrictions for commodity c, which is the set of all links (a, b) that a commodity c packet is allowed to use [3]. Obviously, if there is no routing restriction for commodity c packets, then $\mathcal{L}_c = \mathcal{L}$. The link capacity $\mu_{ab}(t)$ for link (a, b) is defined to be the maximum number of packets that can be transmitted over link (a, b) in one timeslot[4]. In general, multiple commodities might be transmitted over this link during a single timeslot, but the total rate cannot exceed the link capacity $\mu_{ab}(t)$.

Each node i maintains a set of *internal queues* for storing network layer packets according to their commodity. Therefore, the internal queues are also known as *per-destination queues*. In the remainder of this chapter, we will use *internal queues* and *per-destination queues* interchangeably.

Let $A_n^{(c)}(t)$ represent the *cumulative* amount of new commodity c packets that exogenously arrives to source node n by timeslot t (since time 0). Assume these arrival processes are *admissible*. Let $D_{ab}^{(c)}(t)$ be the *cumulative* amount of commodity c packets sent from node a to node b via link (a, b) by timeslot t (since time 0), $a, b, c = 1, \ldots, N$.

Let $Q_n^{(c)}$ denote the internal queue in node n that stores packets destined for node c. With a slight abuse of notation, let $Q_n^{(c)}(t)$ also represent the current backlog of commodity c packets stored in an internal queue at node n. The queue backlog $Q_n^{(c)}(t)$ contains packets that arrived exogenously by $A_n^{(c)}(t)$ as well as packets that arrived endogenously from other nodes by $D_{an}^{(c)}(t)$, $a = 1, \ldots, N$. We define $Q_c^{(c)}(t) = 0$ and $D_{cn}^{(c)}(t) = 0$ for all t, $c = 1, \ldots, N$ and $n = 1, \ldots, N$, so that any packet that has been delivered to its destination is assumed to exit the network right away. The queue backlogs then satisfy the following equation for

[3]As we shall see in Sections 3 and 4, our routing algorithms restrict routing choices to only shortest-path output links when the traffic is not heavy to minimize end-to-end delay. These routing restrictions are denoted by $\mathcal{L}_c$.

[4]Although we define $\mu_{ab}(t)$ here in terms of number of packets, our algorithms and results are applicable to any unit of data as appropriate for the intended application (e.g., the unit of data can just be bits or a rate).

all $n = 1, \ldots, N$ and $c = 1, \ldots, N$ such that $n \neq c$.

$$Q_n^{(c)}(t) = Q_n^{(c)}(0) - \sum_{b=1}^{N} D_{nb}^{(c)}(t) + \sum_{a=1}^{N} D_{an}^{(c)}(t) + A_n^{(c)}(t) \tag{2.1}$$

2.2.2 The Backpressure Algorithm

The original backpressure algorithm was first introduced in [78] in the context of wireless radio networks. It has been shown to achieve optimal throughput [78] and can serve as a solution to certain multi-commodity flow problems [10].

For each link (a, b), the algorithm defines the *optimal commodity* $c_{ab}^*(t)$ as the commodity that maximizes the differential backlog (ties broken arbitrarily):

$$c_{ab}^*(t) \triangleq \arg \max_{\{c | (a,b) \in \mathcal{L}_c\}} \left[Q_a^{(c)}(t) - Q_b^{(c)}(t) \right], \tag{2.2}$$

and defines $W_{ab}^*(t)$ as the corresponding optimal weight:

$$W_{ab}^*(t) \triangleq \max \left[Q_a^{(c_{ab}^*(t))}(t) - Q_b^{(c_{ab}^*(t))}(t), 0 \right]. \tag{2.3}$$

If $W_{ab}^*(t) > 0$, then the internal commodity $c_{ab}^*(t)$ queue is scheduled to be served, and the packets will be transmitted over link (a, b) during timeslot t. Otherwise, no packets will be transmitted over link (a, b) during timeslot t.

It is common in wireless networks that only a subset of all links, referred to as a *schedule*, can transmit packets simultaneously due to interference. Let $\mathcal{S}$ be the set of all possible schedules. The original backpressure algorithm finds the *optimal schedule*, $S^*(t) \in \mathcal{S}$ as an optimization problem as follows:

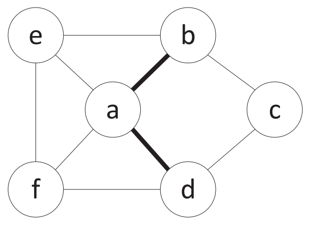
$$S^*(t) = \arg \max_{S \in \mathcal{S}} \sum_{(a,b) \in S} W_{ab}^*(t) \mu_{ab}(t) \tag{2.4}$$

At each timeslot t, for each link $(a, b) \in S^*(t)$, $\mu_{ab}(t)$ packets are removed from $Q_a^{(c^*_{ab})}$ and transmitted to $Q_b^{(c^*_{ab})}$. If $Q_a^{(c^*_{ab})}$ does not have $\mu_{ab}(t)$ packets, then all packets will leave $Q_a^{(c^*_{ab})}$. For *wireline* networks, $\mu_{ab}(t)$ is a typically constant (e.g., one packet per timeslot), and $\mathcal{S}$ is always the set of all links since all links can be activated without interfering with each other.

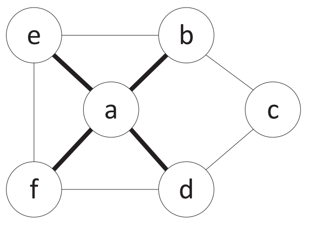
The intuition behind the backpressure algorithm is that packets may not be transmitted if the differential backlog is non-positive, which indicates a congestion at the downstream node. The original backpressure algorithm considers $\mathcal{L}_c = \mathcal{L}$ for any commodity c packet. That is, any packet in node a, no matter what commodity it belongs to, can transmit to *any* neighbor node of node a, as long as the optimal weight computed by Equation 2.3 is positive. This feature essentially exploits all feasible paths in the network for any commodity packet, and as a result, stabilizes the network under heavy traffic loads. However, this feature also incurs large end-to-end packet delays when the network is only lightly or moderately loaded because packets unnecessarily explore and traverse long paths.

In the following sections, we will use the original backpressure algorithm as a baseline algorithm to compare with our proposed algorithms.

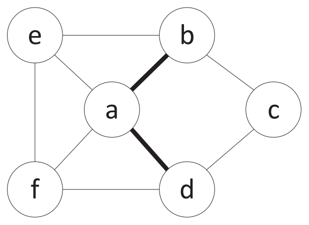
2.3 Route-Expanding BP Algorithms



(a) Queue $Q_a^{(c)}$ is initially in Phase I.



(b) Queue $Q_a^{(c)}$ switches over to Phase II.



(c) Queue $Q_a^{(c)}$ returns back to Phase I.

Figure 2.1: A simple network showing how $\mathcal{L}_c$ (thick edges) changes.

As mentioned in Section 2.2.2, the original backpressure algorithm assumes that $\mathcal{L}_c$

contains all of the links of the network, $\mathcal{L}$. This unconstrained routing may introduce large delays when the traffic load is light, as a packet can unnecessarily explore long paths.

One way to reduce the end-to-end delay is to restrict $\mathcal{L}_c$ to only shortest paths. We call this Shortest-Path Backpressure algorithm (SPBP). Take the network shown in Fig. 2.1(a) as an example, node c has two neighbors, b and d. Neighbor d is on the shortest path for commodity f packets, while neighbor b is not. In the case of SPBP, commodity f packets in node c can only transmit to node d on condition that f is the optimal commodity for link (c, d) and its weight, computed by Equation 3, is positive. In comparison, the original backpressure algorithm allows commodity f packets to transmit to both node b and d, as long as f is the optimal commodity for each link and the weight is positive.

While SPBP can reduce the delay, it also shrinks the network stability region, as it limits the routing choices compared with the original backpressure algorithm. On the other hand, our route-expanding backpressure algorithms can retain the same stability region as the original backpressure algorithm and reduce delay for light or moderate traffic loads by incremental expansion of routing choices. It starts with the shortest-path routing choices as described above for SPBP. To overcome its shortcomings, a dynamic change of routing choices is introduced.

In particular, each internal queue in a node n has two phases of routing. A queue in Phase I can switch over to Phase II when a *transition criterion* is satisfied. A Phase II queue can also return Phase I when the transition criterion is no longer met. Similar to the SPBP, packets in a Phase I queue can only go to a subset of the neighbor nodes, which are on the shortest paths from current node to the destination. In Phase II, similar to the original backpressure algorithm, packets in that queue can be transmitted to *any* neighbor of the current node. The rest of the backpressure scheduling rules are the same, following Equation 2.2, 2.3, 2.4.

The routing choices $\mathcal{L}_c$ are changing dynamically. In the beginning, all internal per-destination queues are in Phase I. This is equivalent to restricting $\mathcal{L}_c$ to allow only links on the shortest paths. When a transition criterion is satisfied, queue $Q_n^{(c)}$ switches over to Phase II, and we add all (n, k) to set $\mathcal{L}_c$ for any neighbor k of node n. When the transition criterion is no longer satisfied, $Q_n^{(c)}$ returns back to Phase I, and we remove those added links from $\mathcal{L}_c$. When all queues in the network are in Phase II, the $\mathcal{L}_c$ becomes $\mathcal{L}$, and this is equivalent to the original backpressure algorithm.

Consider the network shown in Fig. 2.1 as an example and consider queue $Q_a^{(c)}$. In the beginning, $(a, b), (a, d) \in \mathcal{L}_c$, because node b and node d are on shortest paths to node c. When $Q_a^{(c)}$ switches over to Phase II, $(a, e), (a, f)$ are added to $\mathcal{L}_c$. When $Q_a^{(c)}$ returns back to Phase I, $(a, e), (a, f)$ are then removed from $\mathcal{L}_c$.

We propose two transition criteria, a length-based criterion and an age-based criterion, which we refer to the corresponding algorithms as L-BP and A-BP, respectively:

- **L-BP**: Let L_{max} be the maximum backlog that a queue $Q_n^{(c)}$ can stay in Phase I. Whenever $Q_n^{(c)}(t) > L_{max}$, the queue $Q_n^{(c)}$ switches over to Phase II. Whenever $Q_n^{(c)}(t) \leq (1 - \epsilon)L_{max}$, it returns back to Phase I. ϵ can be chosen to provide a *safe margin* between the transition thresholds to prevent a per-destination queue from transitioning back and forth frequently between the two phases (ϵ can be any number in $[0, 1)$; e.g., in the evaluation section, we use $\epsilon = 0$).

- **A-BP**: Consider the head packet of queue $Q_n^{(c)}$. Let $E_n^{(c)}(t)$ represent the *age* of the head packet, which is the period from the timeslot that the head packet enters the queue until current timeslot t. Let A_{max} be the maximum *age* of the head packet for its queue to stay in Phase I. Whenever $E_n^{(c)}(t) > A_{max}$, the queue switches over to Phase II. Whenever $E_n^{(c)}(t) \leq (1 - \epsilon)A_{max}$, it returns back to Phase I. Like L-BP, ϵ can be chosen to provide a safe margin between the transition thresholds to prevent a per-destination queue from transitioning back and forth frequently between the two phases. In the evaluation section, we use $\epsilon = 0$.

2.4 Semi-Oblivious BP Algorithms

In this section, we introduce two semi-oblivious backpressure-based routing algorithms called O-BP and E-BP. Like L-BP and A-BP, the nodes in the network have two phases of routing.

In Phase I, nodes obliviously forward packets using OSPF (O-BP) or ECMP (E-BP). In OSPF [59], a shortest-path next-hop is identified for each destination, and incoming packets are obliviously forwarded to this shortest-path next-hop. In ECMP [37], all equal-cost shortest-path next-hops are identified for each destination, and incoming packets are obliviously routed to one of these shortest-path next-hops with equal probability.

When congestion is detected by monitoring output port queue lengths, a node transitions to Phase II in which the node will start forwarding packets based on the original backpressure routing policy, as described in Section 2.2, in which packets can be forwarded to any next-hop, not just shortest-path next-hops. A node remains in Phase II until the alleviation of congestion is detected, which can also be detected by monitoring output port queue lengths. A key benefit of using O-BP or E-BP over L-BP/A-BP is that the number of backlog information exchanges and differential backlog calculations is largely avoided (packets are simply routed obliviously to a shortest-path next-hop) when O-BP and E-BP are in Phase I. In contrast, L-BP and A-BP require backlog information exchanges and differential backlog calculations for all per-destination queues between the current node and those neighbors which are on a shortest path of a packet. These information exchanges and calculations can be substantial.

However, a major challenge in combining OSPF or ECMP with backpressure routing is how packets are stored in a router. For OSPF, packets are queued at the output port that corresponds to a shortest path and served on a first-come-first-serve basis. For ECMP, packets are queued at multiple output ports that correspond to multiple shortest-path next-hops (if there is more than one shortest path), and packets are served at these output ports on a first-come-first-serve basis. As a result, one output port queue will likely have a mixture packets with many different destinations. However, in backpressure routing, packets are stored in per-destination queues according to their destination rather than their next hop (all packets in the same per-destination queue must have the same destination).

To overcome this incompatibility, we extend a router to implement both output port queues used in OSPF/ECMP and per-destination queues used in the backpressure policy. However, these output port queues and per-destination queues are simply implemented as doubly linked lists with *pointers* to packets rather than storing the actual packet themselves. This way, each packet is still only *physically* stored once. That is, $Q_n^{(c)}$ implements a queue of pointers to packets in node n destined for node c. For each output port, let $Z_n^{(j)}$ denote the output queue in node n that stores pointers to packets waiting at output j. With a slight abuse of notation, let $Z_h^{(j)}(t)$ also represent the current backlog of packets waiting at output j.

When a router receives a packet, it is linked to the tail of its corresponding per-destination queue and an output port queue. For O-BP, the choice of output port queue corresponds to the shortest-path output as determined by OSPF. For E-BP, one shortest-path output port is selected with equal probability. When a router is in Phase I, a packet will depart from the head of each output port queue, on the condition that the differential backlog for the commodity that this packet belongs to is positive, in which case the corresponding entry for that packet in the per-destination queue will be removed. Otherwise, if the differential backlog for the commodity that the packet at the head of an output port queue is non-positive, no packet will be moved. This ensures that our algorithms will never move a packet if the differential backlog is non-positive no matter whether the router is in Phase I or Phase II, which is required for the throughput optimality proof in Appendix A.1.

When a router is in Phase II, the router will use the per-destination queue information of its neighbors to compute the backpressure weight, then choose the packet from the per-destination queue with the maximum positive weight, and forward it to its neighbor accordingly, and the corresponding entry in the output port queue will also be removed.

More specifically, the transition criteria for O-BP and E-BP are as follows:

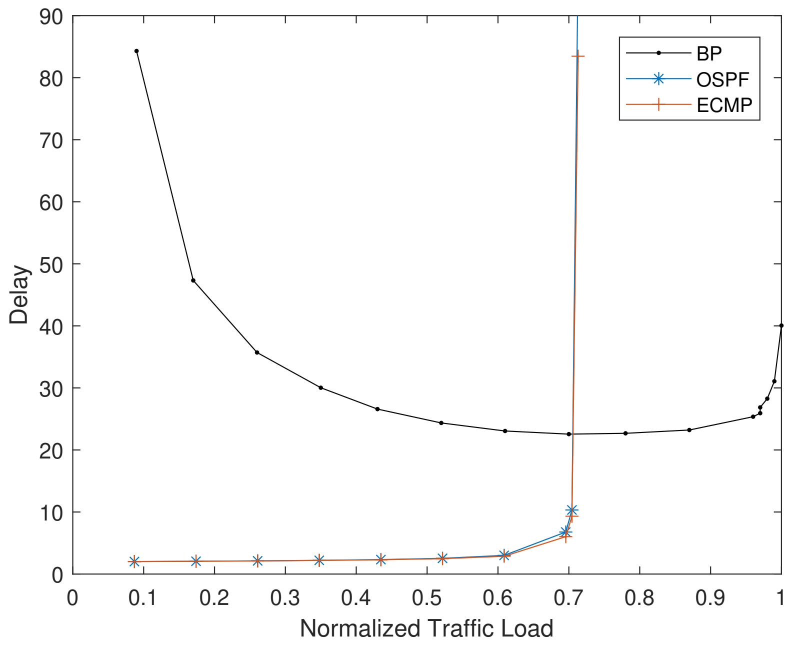
- **O-BP** and **E-BP**: Let H_{max} be the maximum backlog that any output port queue $Z_n^{(j)}$ can have for a router to stay in Phase I. Whenever $Z_n^{(j)}(t) > H_{max}$ for *any* output j, the *entire* router switches over to Phase II. Whenever $Z_n^{(j)}(t) \leq (1-\epsilon)H_{max}$ is satisfied for *all* outputs, the entire router returns back to Phase I. ϵ can be chosen to provide a safe margin between the transition thresholds to prevent a router from transitioning back and forth frequently between the two phases (ϵ can be any number in $[0, 1)$; e.g., in the evaluation section, we use $\epsilon = 0$).

2.5 Evaluations

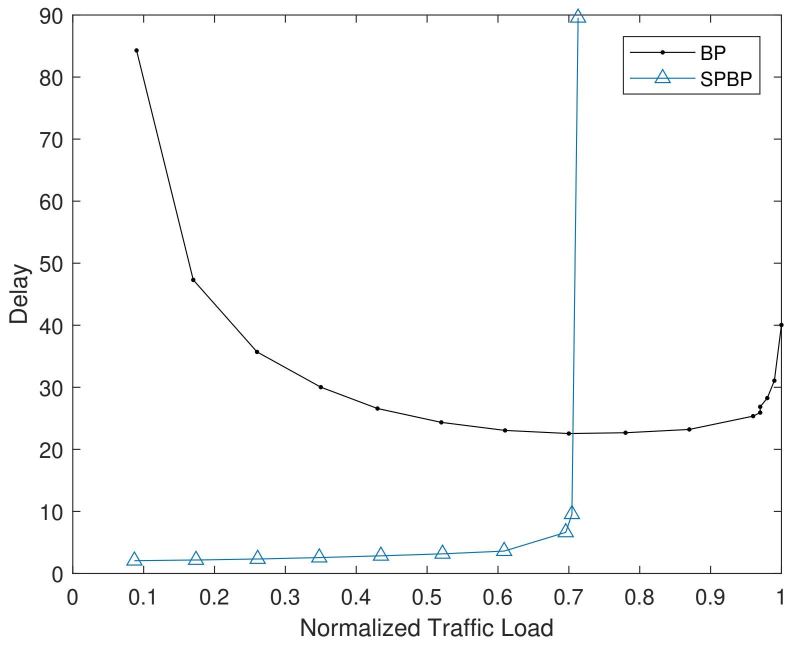
In this section, we present evaluations of our proposed route-expanding and semi-oblivious algorithms. To evaluate these algorithms, we focus on the adaptive routing problem for the wireline case. In particular, we present our evaluations using a real, large PoP-level backbone network, namely the Abilene[40] network. The Abilene network has been studied and discussed in the research literature. Its network topology, traffic dataset, and routing information are available in the public domain [90]. In the following, we first describe our experimental setup and then present our simulation results.

2.5.1 Experimental Setup

The Abilene network is a public academic network in the U.S. with 12 nodes interconnected by OC192, 9.92 Gbits/s links. We use the traffic matrices obtained in [90] in the experiments. Each traffic matrix consists of the demand rate of every source

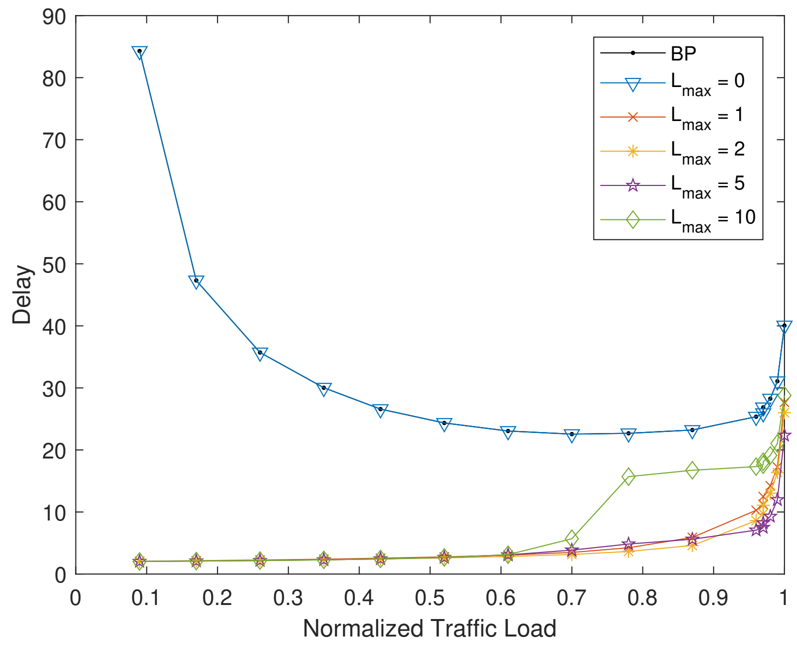


(a) BP vs. OSPF/ECMP

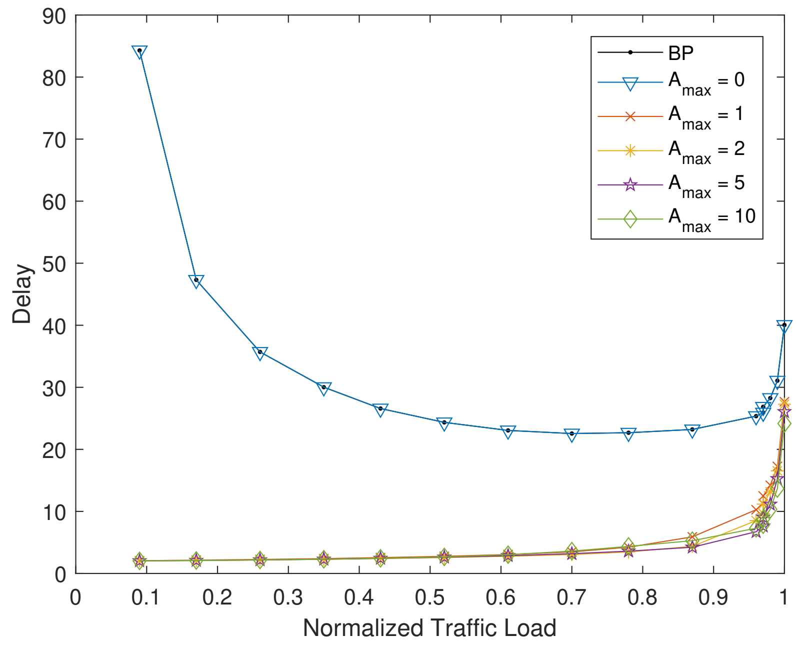


(b) BP vs. SPBP

Figure 2.2: Delay comparison under different traffic loads for shortest-path routing.

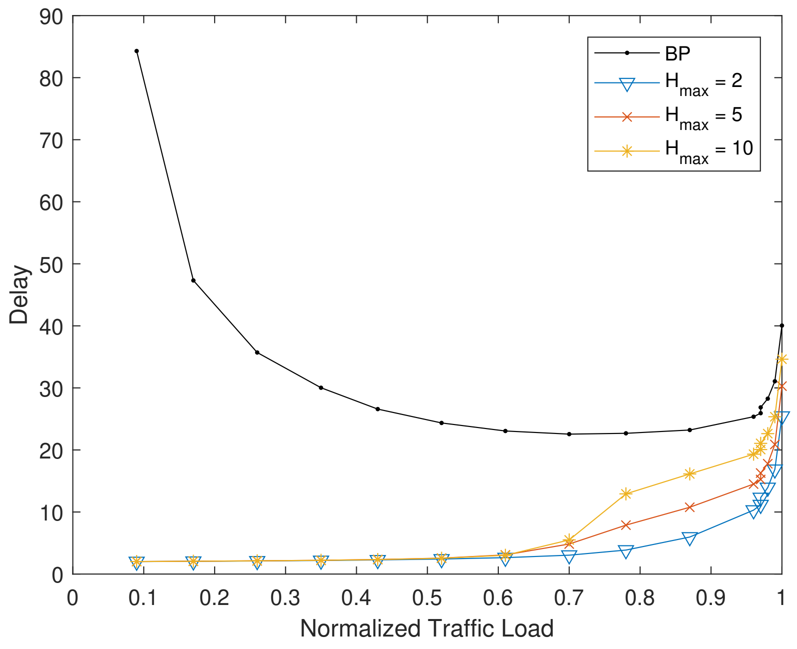


(a) BP vs. L-BP

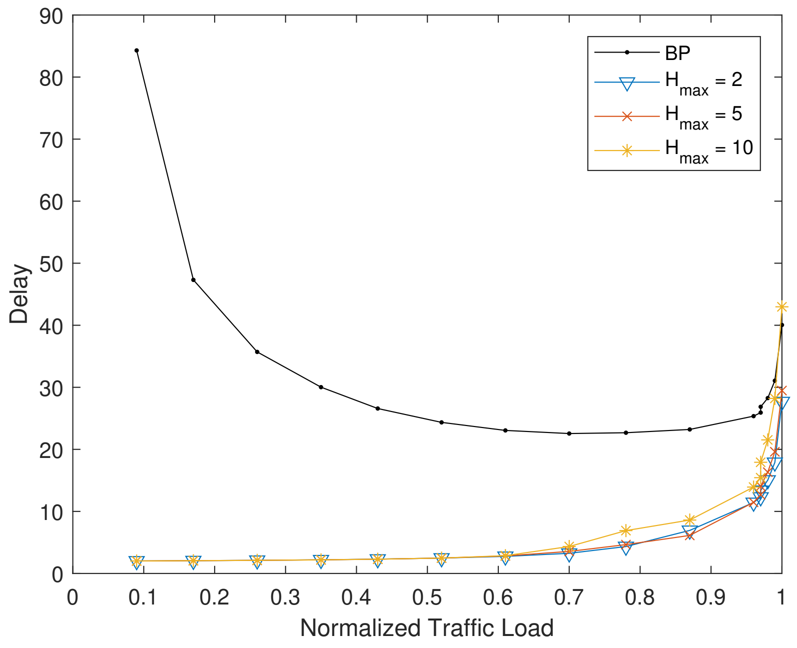


(b) BP vs. A-BP

Figure 2.3: Delay comparison under different traffic loads for route-expanding backpressure routing.



(a) BP vs. O-BP



(b) BP vs. E-BP

Figure 2.4: Delay comparison under different traffic loads for semi-oblivious backpressure routing.

destination pair within five minutes. Therefore, these traffic matrices provide a snapshot of real total demand offerings between each source-destination pair in the Abilene network every five minutes. The actual dataset spans from March 1, 2004 to September 4, 2004. As the traffic matrices indicate, the Abilene network is underutilized. To demonstrate that our new backpressure-based algorithm improves delay performance while retaining optimal throughput, we selected the traffic matrix with the highest traffic load, and scaled it by different factors. We incrementally increased the scaling factor until the resulting arrival rates exceed the network's stability region. Then we normalized that largest scaling factor.

We implemented our simulator in C++. Traffic generation follows a Bernoulli arrival process with probability

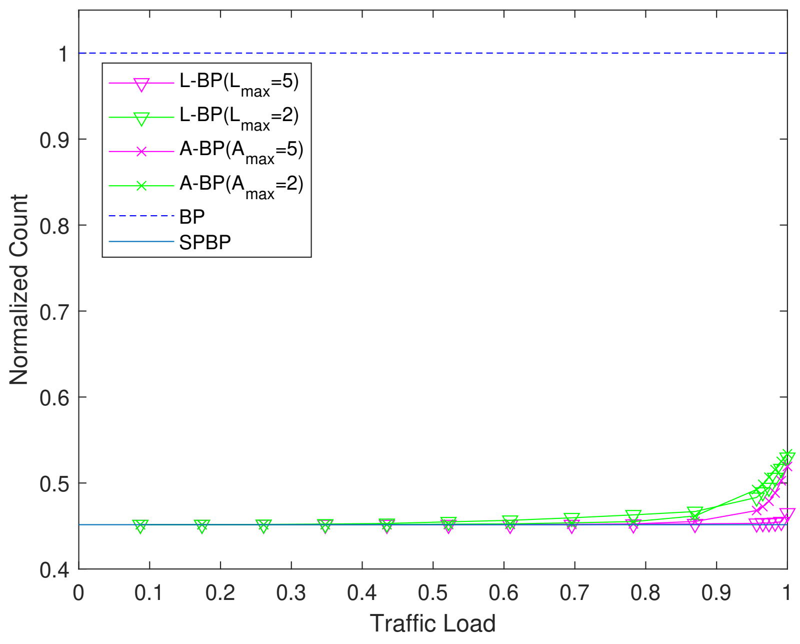
$$p = \frac{\text{traffic demand}}{\text{link capacity}}.$$

We assume $\mu_{ab}(t) = 1$. That is, at each timeslot, at most one packet may be transmitted over each link. The end-to-end delay is measured by the time period from the timeslot when a packet enters the network by the traffic generation function to the timeslot when the packet arrives at its destination and thus leaves the network. To get reliable results, the simulation time should be long enough for the network to reach a steady state. In particular, for each scaling factor, we simulated 40 million timeslots, of which the first 20 million timeslots serve as a warm-up phase. We then collect 200 data points by sampling every 100,000 timeslots for the remaining 20 million timeslots. Finally, we average over these 200 data points for the results presented in this section for each scaling factor.

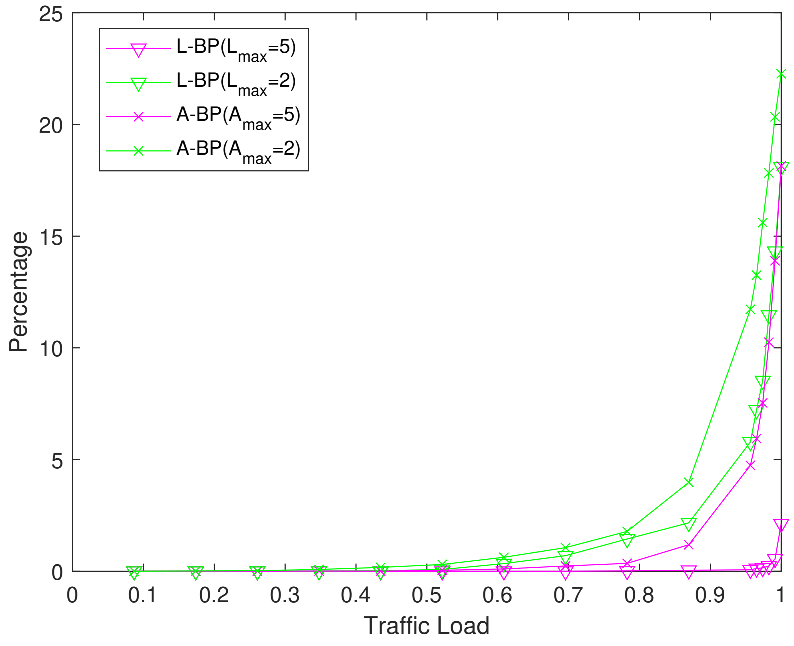
2.5.2 Experimental Results

Delay Performance

In this section, we present and compare simulaton results for end-to-end delay performance for the original backpressure algorithm (BP), Open-Shortest-Path-First (OSPF)



(a) Backlog information exchange frequency.



(b) Percentages of Phase II queues.

Figure 2.5: Backlog information exchange frequency and percentages of Phase II queues for L-BP and A-BP.

routing, Equal-Cost Multi-Path (ECMP) routing, Shortest-path backpressure (SPBP) routing, the L-BP and A-BP route-expanding BP routing algorithms, and the O-BP and E-BP semi-oblivious BP routing algorithms. The results are shown in Fig. 2.2, Fig. 2.3 and Fig. 2.4. From these results, we can observe the following:

- **Original BP**: We observe in Fig. 2.2 that under the original backpressure algorithm, the delay first decreases and then increases with increasing traffic loads. This phenomenon validates that the original backpressure algorithm incurs large delays under light or moderate traffic loads, because packets may explore unnecessarily long paths.

- **OSPF and ECMP**: From Fig. 2.2(a), we observe that OSPF and ECMP have low average end-to-end delays compared with the original backpressure algorithm. However, they both saturate the network early, achieving only about 72% of the achievable throughput.

- **SPBP**: Fig. 2.2(b) illustrates that although SPBP also achieves low average end-to-end delays, it is still not throughput optimal because it saturates the network early, achieving similarly only about 72% of the achievable throughput. This is to be expected since reducing routing choices shrinks the network stability region.

- **L-BP**: Fig. 2.3(a) shows that L-BP route-expanding backpressure algorithm can achieve the same optimal throughput as the original backpressure algorithm and how the choices of the threshold L_{max} impact the delay performance. As expected, when $L_{max} = 0$, the delay performance is exactly the same as the original backpressure algorithm, because all per-destination queues are always in Phase II. Our experiments show that as long as $L_{max} \geq 1$, the L-BP algorithm is able to reduce the delay by more than 80% compared with the original backpressure algorithm when the traffic load is light or moderate. To obtain better delay performance, L_{max} should be carefully chosen and tuned. Intuitively, L_{max} should not be too big, as a big L_{max} may impede the entrance into Phase II and the accumulated packets may have larger queuing delays than those in a smaller L_{max} setting. For example, as illustrated in Fig. 2.3(a), the delay for $L_{max} = 10$ increases significantly at higher traffic loads, because those congested per-destination queues do not enter Phase II in time. On the other hand, if L_{max} is too small, then it would be too easy for the per-destination queues to enter Phase II and route the packets unnecessarily to longer paths.

- **A-BP**: Fig. 2.3(b) shows that A-BP route-expanding backpressure algorithm can achieve the same optimal throughput as the original backpressure algorithm and how the choices of the threshold A_{max} impact the delay performance. Similar to L-BP, when $A_{max} = 0$, the delay performance is exactly the same as the original backpressure algorithm, because all per-destination queues are always in Phase II. In addition, our experiments show that as long as $A_{max} \geq 1$, the A-BP algorithm is able to reduce the delay by more than 90% compared with the original backpressure algorithm when the traffic load is light or moderate. The parameter A_{max} should also be tuned carefully. For the same reason, A_{max} should not be too big, as a big A_{max} may prevent the queue from entering Phase II in time. Also, if the threshold is too small (e.g. $A_{max} = 1$), then it makes the queues too easy to enter Phase II.

- **O-BP and E-BP**:

  Fig. 2.4 shows that, similar to L-BP and A-BP, the O-BP and E-BP semi-oblivious backpressure algorithms can also both achieve the same optimal throughput as the original backpressure algorithm. Our experiments show that when $H_{max} \geq 2$, both O-BP and E-BP are able to reduce the delay at light or moderate traffic dramatically. As before, H_{max} should be chosen carefully to achieve better performance. In general, smaller H_{max} is more favorable. In our experiments, $H_{max} = 2$ has the best delay performance for both O-BP and E-BP algorithms.

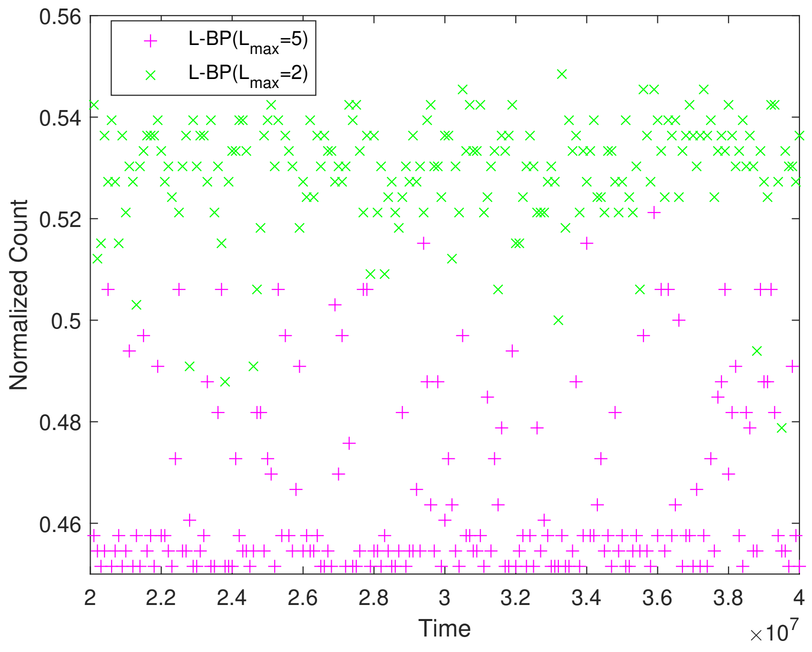
Backlog Information Exchanges in L-BP and A-BP

In this section, we examine two metrics: the *backlog information exchange frequency* and the *percentage of queues* in the network that are in Phase II for L-BP and A-BP.

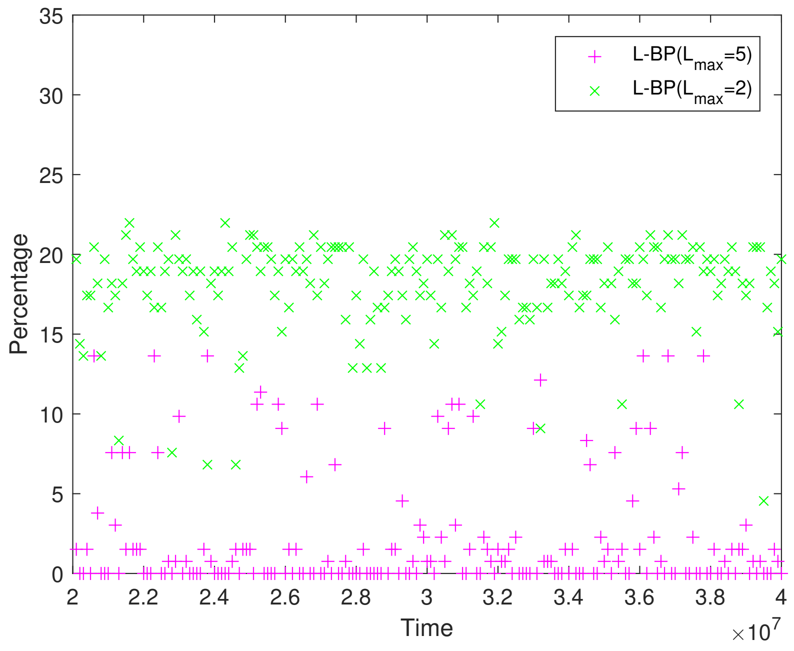
A *backlog information exchange* is recognized when an internal per-destination queue needs its neighbor's corresponding per-destination queue backlog information to compute the backpressure. For example, according to the original backpressure algorithm, if a node has three neighbors, then each internal per-destination queue has to know the backlog information from its three neighbors, and the corresponding differential backlogs need to be computed for these three neighbors. However, in our proposed algorithms, if a queue is in Phase I, then the backlog information only needs to be exchanged with a subset of neighbor nodes that are a part of some shortest path routes, and the differential backlog calculations only need to be computed with respect to these nodes.

It should be noted that, among the original BP algorithm, SPBP, L-BP, and A-BP, the original BP algorithm needs the maximum number of backlog information exchanges, which is the upper bound. SPBP needs the least, which is the lower bound. For the L-BP/A-BP algorithms, whether their backlog information exchange frequencies are closer to BP or SPBP depends on how many queues in L-BP/A-BP are in Phase II. If no queue in L-BP/A-BP is in Phase II, then L-BP/A-BP is the same as SPBP. If all queues in L-BP/A-BP are in Phase II, then L-BP/A-BP is the same as BP.

Compared with SPBP, which is *not* throughput optimal, L-BP/A-BP adaptively transitions per-destination queues to Phase II from Phase I *as needed* to achieve the optimal throughput. However, the more per-destination queues are transitioned into Phase II, the higher is the amount of backlog information that a node has to exchange with its neighbors. Fig. 2.5 shows that the increase in backlog information exchange is very small, which is a

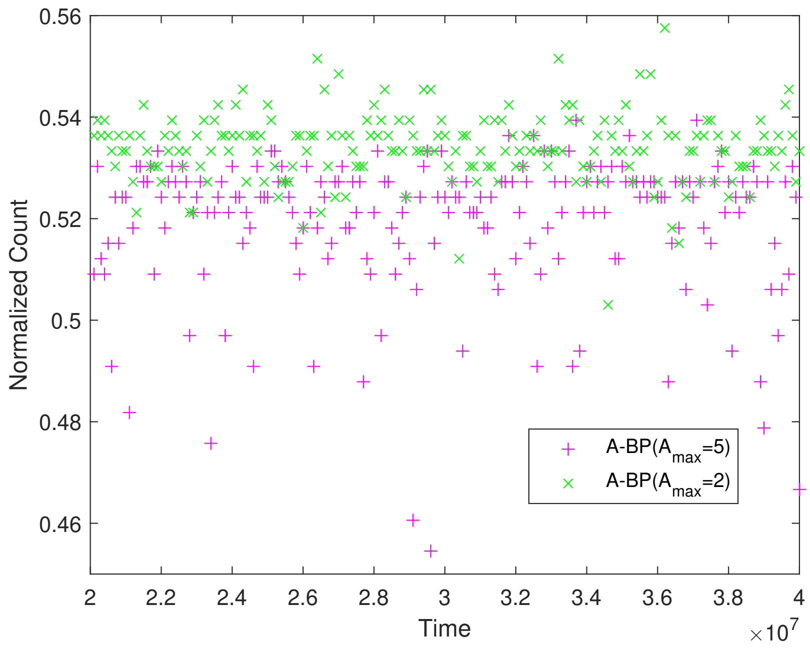


(a) Backlog information exchange frequency.

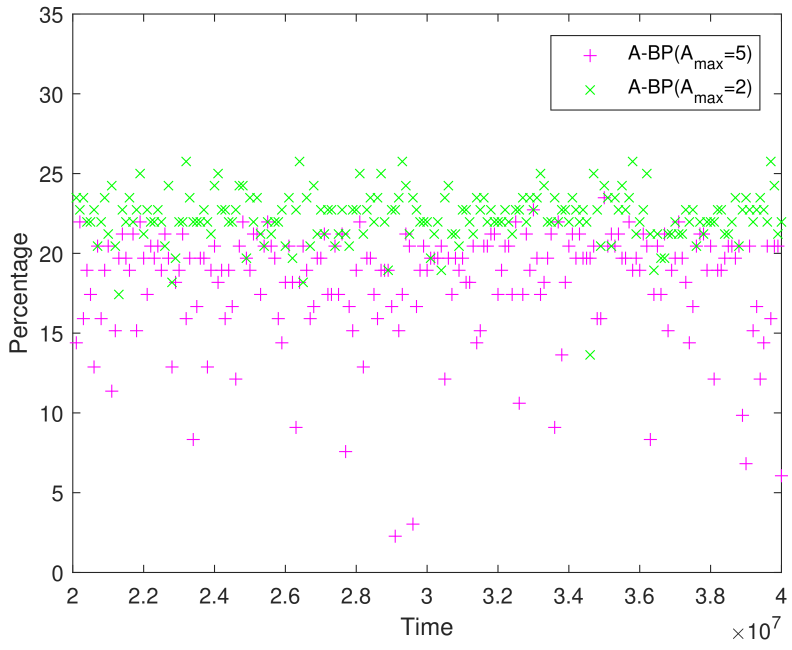


(b) Percentages of Phase II queues.

Figure 2.6: Backlog information exchange frequency and percentages of Phase II queues for L-BP under maximum traffic load.



(a) Backlog information exchange frequency.



(b) Percentages of Phase II queues.

Figure 2.7: Backlog information exchange frequency and percentages of Phase II queues for A-BP under maximum traffic load.

small price to pay for optimal throughput.

In particular, Fig. 2.5(a) shows the backlog information exchange frequency for BP, SPBP, L-BP, and A-BP algorithms, under different traffic loads. The information exchange frequency is normalized to the number of backlog exchanges required by BP, which has the highest frequency as BP requires information exchanges with all neighbors. On the other hand, SPBP has the lowest frequency as it only requires information exchanges with shortest-path neighbors; the normalized frequency for SPBP is about 0.45. Surprisingly, L-BP and A-BP only require slightly more backlog information exchanges compared with SPBP, even at very high traffic loads. This means that, by adding a bit more backlog information exchanges, L-BP and A-BP can achieve optimal throughput, which is a much higher throughput than SPBP. In Fig. 2.5(b), we can see an increasing number of queues are switched over to Phase II when the traffic load increases. For the same group of algorithms, L-BP or A-BP, the smaller the threshold is, the more queues are switched over to Phase II as the the traffic load increases, because a smaller threshold makes the router easier to switch over the Phase II. Nevertheless, the increase is negligible if the threshold and algorithm is properly chosen. For example, the percentage of Phase II routers for L-BP ($L_{max} = 5$), even under the highest traffic load, is still less than 3%. For others, even under very high traffic loads, the percentages are still below 23%. All of this translates to much lower computational requirements for calculating the necessary different backlogs.

Fig. 2.6 and Fig. 2.7 take the maximum traffic load (the traffic load we use in our experiments that is just about to saturate the network) as an example, and show the number of backlog information exchanges and the percentages of Phase II queues for L-BP and A-BP, respectively.

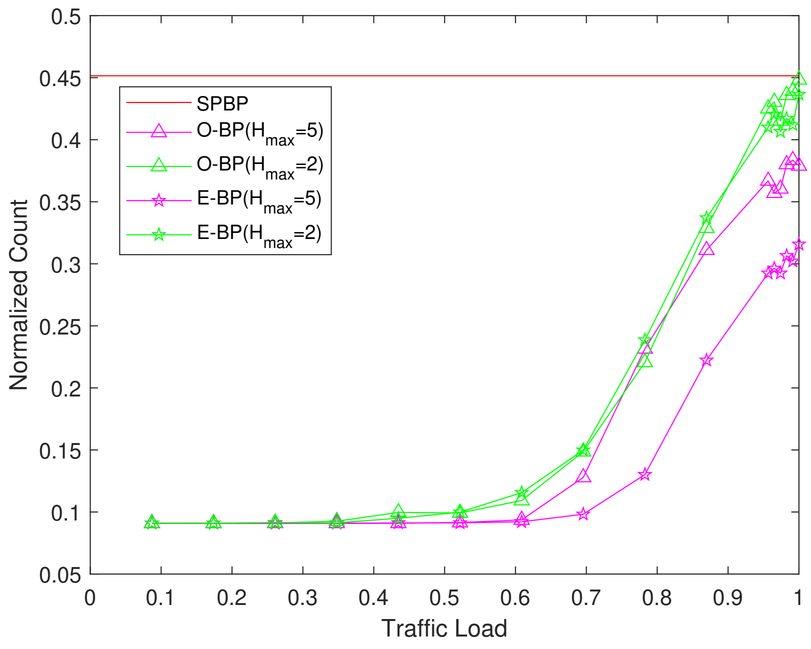
In every figure, we can see that a bigger threshold usually incurs less information exchanges, because less routers are switched over to Phase II. With a carefully chosen threshold, an algorithm may only require the similar amount of backlog information exchanges to SPBP algorithm. For example, in Fig. 2.6(a), most data points from L-BP ($L_{max} = 5$) for the backlog information exchanges are between 0.45 and 0.46. Recall that the lower bound from SPBP for the normalized backlog information exchange frequency is about 0.45. Therefore, Fig. 2.6(a) shows that most of the time the backlog information exchange frequency of L-BP ($L_{max} = 5$) is comparable to the SPBP algorithm. For other algorithms, as can be seen from Fig. 2.6(a) and Fig. 2.7(a), most of the data points are below 0.54, which indicates that most of the time the frequency of the backlog information exchanges is also limited.

Fig. 2.6(b) and Fig. 2.7(b) show that most of the time only less than 25% queues are switched over to Phase II. This indicates that our algorithms are able to identify the most congested areas of the network, switch over those queues to Phase II, and keep the remaining queues working in Phase I.

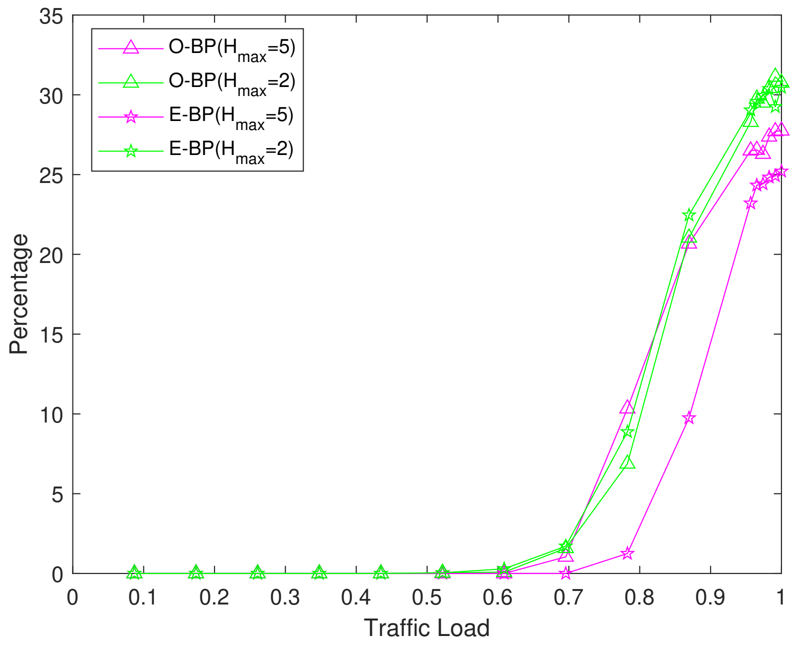
Backlog Information Exchanges in O-BP and E-BP

In this section, we examine the *backlog information exchange frequency* and the *percentage of routers* in the network that are in Phase II for the O-BP and E-BP semi-oblivious BP algorithms.

When a router is in Phase I, for each link, it only needs backlog information from the corresponding neighbor for a *single* commodity, which the packet at the head of the output port queue belongs to. In comparison, when a router is in Phase II, it needs the backlog information from all its neighbors for *all* commodities. Fig. 2.8(a) compares the backlog information exchange frequency among SPBP, O-BP and E-BP algorithms under different traffic loads. As we can see, O-BP and E-BP require substantially lower frequency of backlog information exchanges. This is because when the traffic load is not high, most routers are still in Phase I, and thus they only need very few information exchanges. Even when the traffic load becomes high, O-BP and E-BP still incur less information exchanges



(a) Backlog information exchange frequency.



(b) Percentages of Phase II routers.

Figure 2.8: Backlog information exchange frequency and percentages of Phase II routers for O-BP and E-BP.

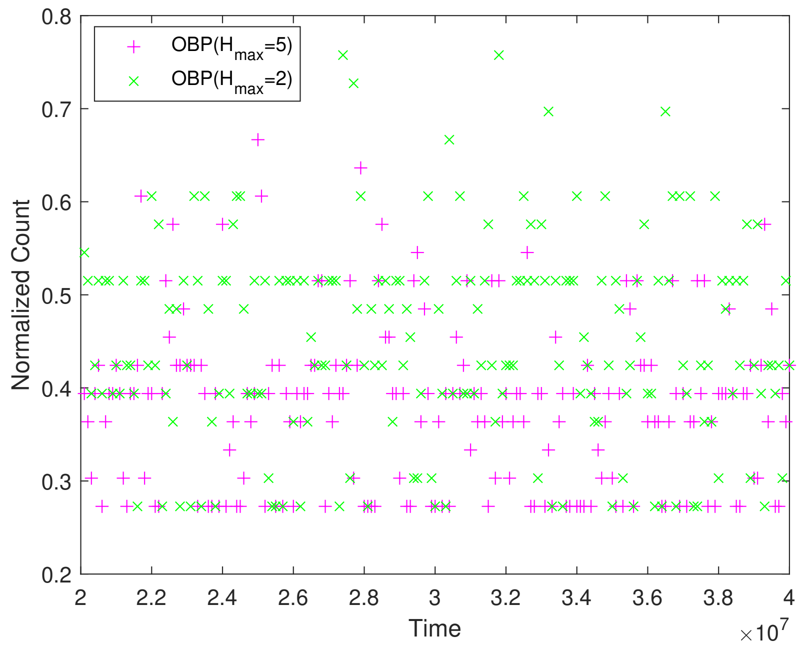
than SPBP. This is because, as shown in Fig. 2.8(b), a large percentage of the routers in the network still remain in Phase I, and these Phase I routers only need a small fraction of backlog information from their neighbors. Although the Phase II routers have to exchange backlog information with its neighbors for all commodities, the number of such Phase II routers is very small. As a result, O-BP and E-BP always require less information exchanges than SPBP for the entire range of traffic loads simulated.

Fig. 2.9(a) and Fig. 2.10(a) show the amount of backlog information exchanges at the maximum traffic load for O-BP and E-BP respectively. Similar to L-BP and A-BP algorithms, a bigger threshold leads to less information exchanges, because more routers are still in Phase I. With a carefully chosen threshold, our algorithm can reduce the information exchanges dramatically. Take $H_{max} = 5$ as an example, even under this maximum traffic load, the normalized amount the information exchanges of either of O-BP or E-BP *rarely* exceeds 0.45, which is even less than the normalized frequency required by SPBP. This is because, as shown in Fig. 2.9(b) and 2.10(b), even when the traffic load is about to saturate the network, there are still about 65% to 75% routers that are in Phase I, which only need a small amount of backlog information from their neighbors. Besides, that the percentage is concentrated at a fixed number of values indicates that only some particular routers are congested in the network at this traffic load. Our algorithms are able to identify the congested areas in the network and only switch over those routers to Phase II.

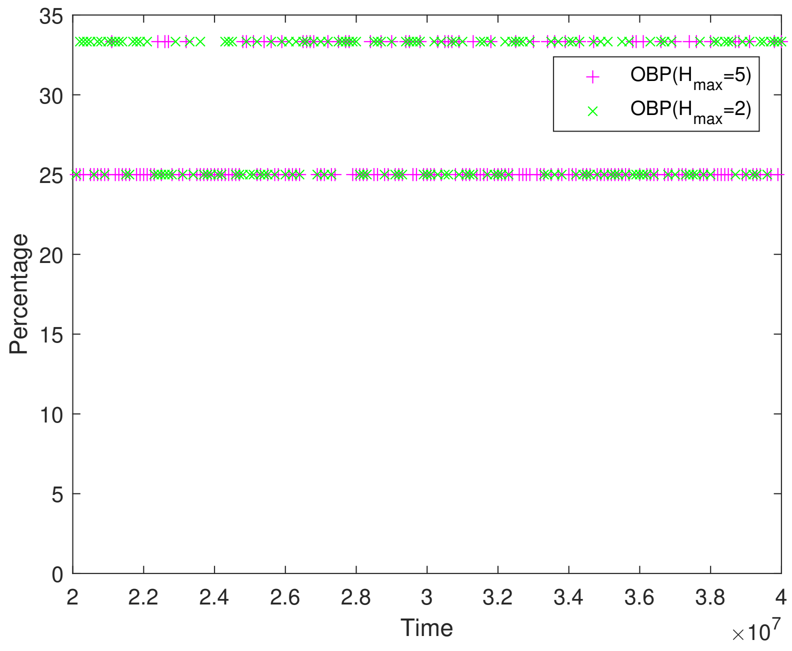
Comparison with prior proposed algorithms

We have implemented the *MinHop* algorithm proposed in [86] to compare with our proposed algorithms in Fig. 2.11.

Fig. 2.11(a) shows that our delay performance is comparable with the *MinHop* algorithm. The delay performance of the MinHop algorithm is sensitive to the choice of the parameter K. If K is too big, then their algorithm will favor shortest paths. As long as

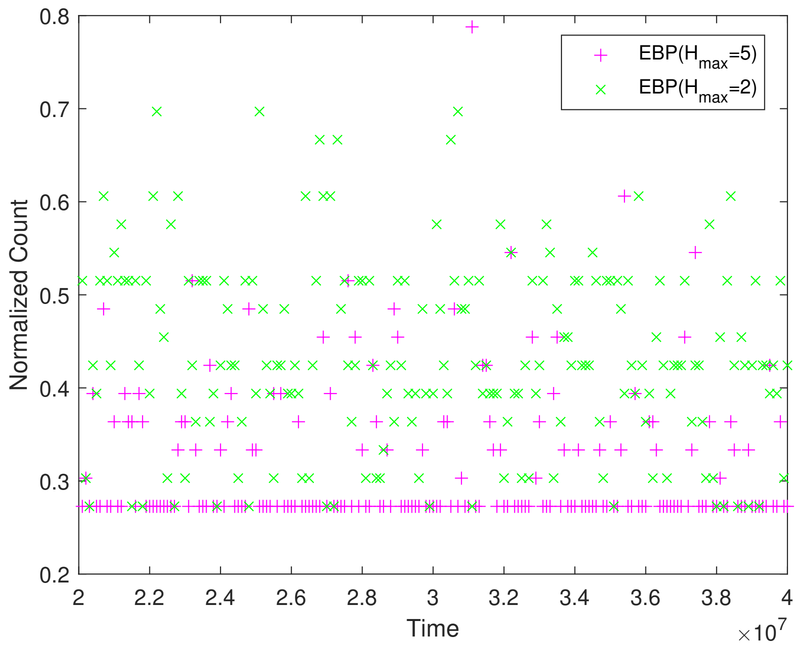


(a) Backlog information exchange frequency.

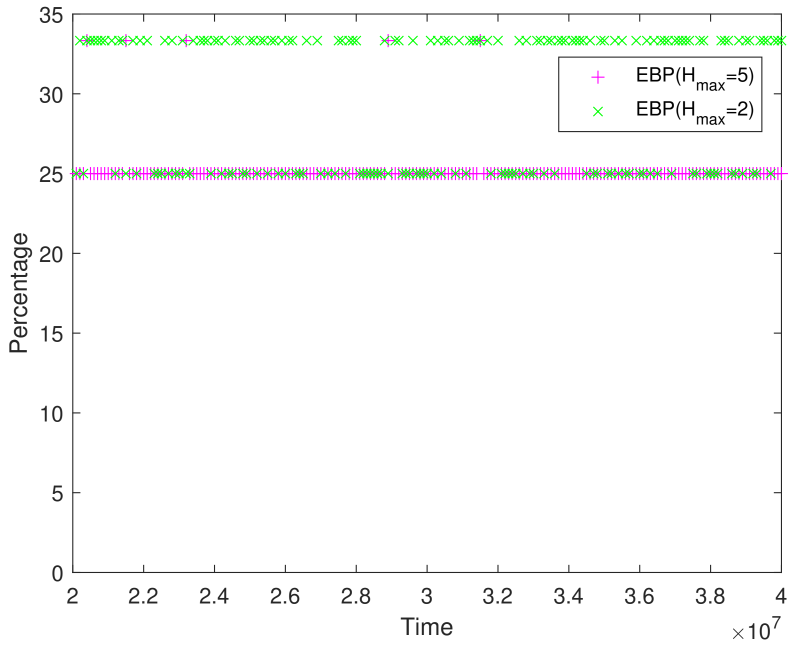


(b) Percentages of Phase II routers.

Figure 2.9: Backlog information exchange frequency and percentages of Phase II routers for O-BP under maximum traffic load.

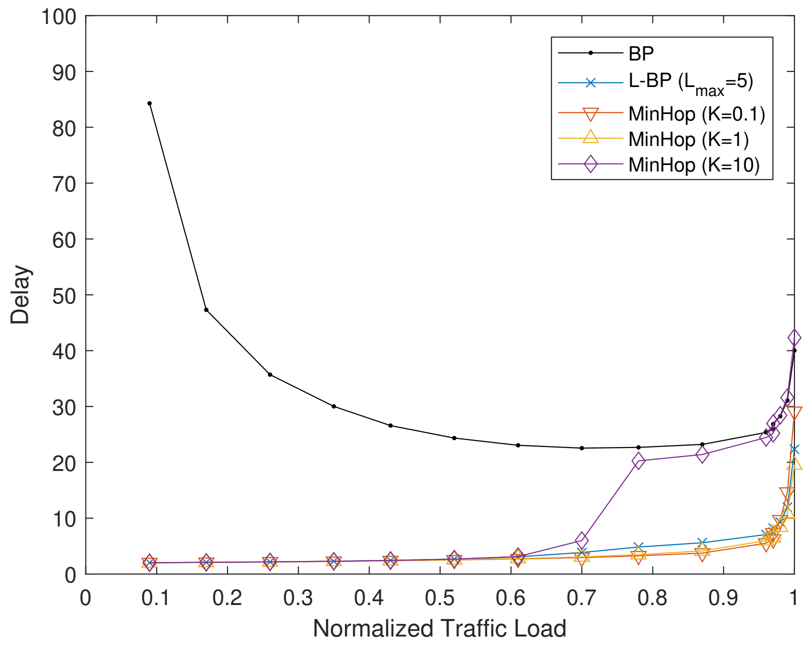


(a) Backlog information exchange frequency.

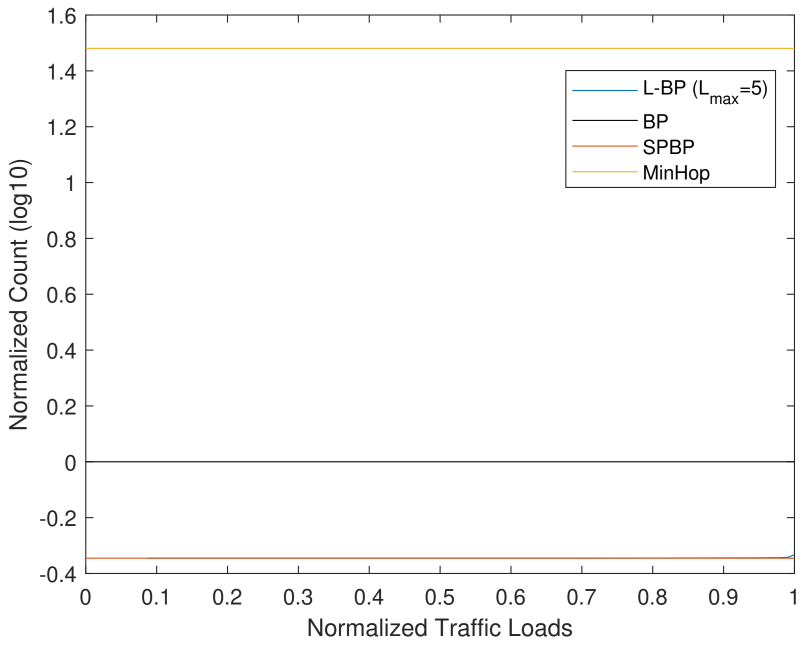


(b) Percentages of Phase II routers.

Figure 2.10: Backlog information exchange frequency and percentages of Phase II routers for E-BP under maximum traffic load.



(a) Delay comparison with [86].



(b) Information exchanges comparison with [86].

Figure 2.11: Compare with [86] (referred here as "MinHop").

the K is not infinite, the throughput still remains optimal, but the delay can be extremely huge. On the other hand, if K is too small, for example $K = 0.1$, then the delay is also compromised, because it just favors shorter queues. As a result, K plays an important role in improving delay performance, which is similar to what our L_{max}, A_{max}, and H_{max} do.

The main drawback of [86] is its complexity. As can be seen from Fig. 2.11(b), due to the huge number of queues that must be maintained per router, the amount of information exchanges is enormous. When compared with even the original backpressure algorithm, the amount of information exchanges required by MinHop [86] is 30 times higher, which in turn is far higher than what our algorithms require.

2.6 Conclusion

In this chapter, we proposed two new route-expanding backpressure-based adaptive routing algorithms, called L-BP and A-BP, that are based on the incremental expansion of routing choices in response to congestion at a node on a per-destination basis. We also proposed two new semi-oblivious backpressure-based adaptive routing algorithms, called O-BP and E-BP, that are based on oblivious routing by default and incremental expansion to backpressure routing on a per-node basis. All variants are shown to be throughput optimal by means of fluid model analysis. Simulation results using actual traffic profiles on a public network demonstrate that our proposed algorithms indeed provide substantial improvements in delay performance. The simulation results further show that in practice, our approach dramatically reduces the amount of differential backlogs that has to be computed and the amount of backlog information that has to be exchanged because routing choices are only incrementally expanded as needed.

2.7 Acknowledgment

Chapter 2, in part, is a reprint of the material as it appears in ACM Transactions on Modeling and Performance Evaluation of Computing Systems, January 2019. Yin, Ping; Yang, Sen; Xu, Jun; Dai, Jim; Lin, Bill. Chapter 3 also includes the materials as it appears in ACM/IEEE Symposium on Architectures for Networking and Communications Systems Conference, 2017. Yin, Ping; Yang, Sen; Xu, Jun; Dai, Jim; Lin, Bill. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Network Optimization for Unified Packet and Circuit Switched Networks

3.1 Introduction

Internet traffic continues to grow unabatedly at a rapid rate, driven largely by more and more video content, from 1080p HD to 4K Ultra HD video streaming today, to 8K Ultra HD video streaming in the near future. Although the packet-switching approach used in Internet backbone networks has thus far been able to keep up, it is unclear whether electronic routers that have been used at the core of backbone networks will continue to scale to match future traffic growth. On the other hand, optical fiber and switching elements have demonstrated an abundance of capacity that appears to be unmatched by electronic routers. The rate of increase in optical transport capacity has been keeping pace with traffic growth. Thus, one way of keeping pace with future traffic demands is to build an all-optical backbone network. However, packet switching requires the buffering of packets, of which optical switches are not capable today, and it appears unlikely that reasonable size packet buffers can ever be practically realized in optics. On the other hand, circuit switching has a much simpler data transport, making it well-suited to optics and its vast capacity potential.

To harness the huge capacity of optical circuit switching in IP networks, researchers have explored different ways of implementing IP over dynamically configurable optical transport networks [22, 12, 88, 71, 52, 58, 24]. These earlier efforts assumed a GMPLS-based control plane [58, 24]. More recently, given the broad success of software-defined networking (SDN) [72, 49, 83, 55], there has been considerable renewed interest in unified packet and circuit switched network architectures based on SDN as the unified control plane [30, 31]. In the SDN-based unified architecture proposed in [31], backbone routers are replaced with less expensive hybrid optical-circuit/electrical-packet switches that have both circuit-switching and packet-switching capabilities. These hybrid switches are logically connected in a fully-meshed network where each hybrid switch implements an IP node, and where each IP node is logically connected to each and every other IP node via a single direct circuit-switched hop. This unified packet and circuit-switched network can then be managed using a single converged control plane.

Figure 3.1 depicts this unified fully-meshed IP network architecture. The actual underlying optical transport network can be dynamically allocated to provide different circuit capacities to implement each logical connection in the full-mesh, for example based on estimated traffic demands. For example, in figure 3.1, a logical connection from San Francisco (SF) to New York (NY) may be implemented as an optical circuit-switched path via Seattle and Chicago. In general, a logical connection may be implemented over multiple physical paths.

There are several key advantages with an SDN-based unified fully-meshed architecture:

- First, studies have shown that up to 85% of the packets that are processed by

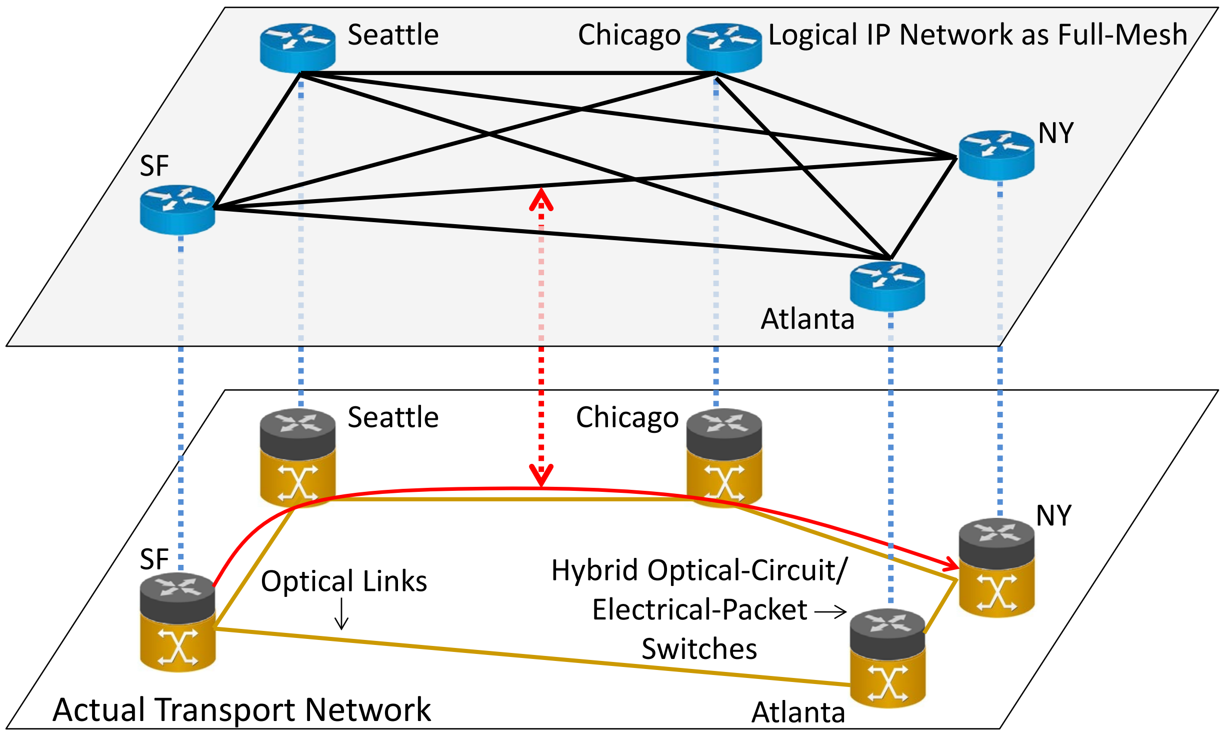


Figure 3.1: IP network logically as a full-mesh, with logical connections implemented over an optical circuit-switched transport network and logical routers implemented as part of hybrid optical-circuit/electrical-packet switches.

backbone routers today are just pass-through traffic [74, 71, 31]. Therefore, packets are unnecessarily delayed due to queuing time at intermediate routers. With a unified architecture, packets can traverse the circuit-switched network through pre-established circuits (light-paths) at optical speeds from the source node to the destination node in a single logical hop.

- Second, backbone routers are unnecessarily expensive today because they must be designed to process all packets, including all pass-through packets. With a unified architecture, expensive packet-switched router ports are primarily needed only for interfacing with access routers; pass-through traffic can be handled by less expensive circuit-switched ports. This approach promises to dramatically reduce capital expenditures [74, 71, 52, 39, 31].

- Finally, a unified architecture is expected to be far more scalable since most traffic can be switched end-to-end using scalable optical transports.

A key problem that must be solved in this unified architecture approach is the allocation of optical circuits between adjacent IP nodes in the logical full-mesh (*i.e.*, between every IE pair of ingress and egress nodes). In this chapter, we propose to formulate our circuit allocation problems as convex optimization problems. In particular, the main contributions of this work are as follows:

- We propose a novel convex optimization framework based on a new destination-based multicommodity flow formulation for the allocation of circuits in unified packet and circuit switched networks.
- We consider two deployment settings for circuit allocation. In the first setting, we consider the case in which real-time traffic measurements are possible, and we can dynamically allocate circuits on a frequent basis in response to changing traffic.
- In the second setting, we consider the case in which we allocate circuits based on historical traffic patterns. Previous studies have shown that the aggregate traffic at the core of the network tends to be very smooth and that it follows strong diurnal patterns [22, 67, 56]. Such diurnal traffic observations over repeated data sets suggest that circuits can be allocated based on historical data. In this setting, circuit configurations can be precomputed offline.
- In both settings, we formulate global network optimization objectives as concave functions that capture the fair sharing of network capacity among competing traffic flows. The convexity of our problem formulations ensures globally optimal solutions.

The rest of this chapter is organized as follows. In §3.2, we present our destination-based multicommodity flow formulation for circuit allocation that reduces the number of decision variables in the convex optimization problems by a factor of n, where n is the number of nodes in the network. In §3.3, we present our formulations of the real-time based and history-based circuit allocation problems as convex optimization problems. We then describe our experimental setup in §3.4, and we present the results of our evaluations in §3.5. In §3.6, we discuss additional related work. Finally, we present concluding remarks in §3.7.

3.2 Destination-Based Multi-Commodity Flow Formulation

We formulate our optical circuit allocation problems as multi-commodity flow optimization problems. We consider a backbone network with n nodes and m directed edges, and we index nodes as $i = 1, \ldots, n$ and edges as $j = 1, \ldots, m$. Note that an undirected edge between nodes k and ℓ can be modeled by two directed edges, one from k to ℓ and the other from ℓ to k. With n nodes, we have $n(n-1)$ ingress-egress (IE) pairs, and we index IE pairs as (k, ℓ), which refers to ingress (source) ℓ and egress (destination) k (*i.e.*, from node ℓ to node k).

Classically, multi-commodity flow formulations typically use $n(n-1)m$ flow assignment variables, each of which defines the fraction of the corresponding IE pair traffic (among $n(n-1)$ IE pairs) along the corresponding edge (among m edges). In this chapter, we propose a *destination-based* multi-commodity flow formulation in which the *flows* ("commodities") are labeled by their destination or egress node k rather than by an IE pair. This reduces the number of flow assignment variables by a factor of $n-1$ to nm variables. This substantial reduction in the number of variables allows us to scale the method to far larger networks. To the best of our knowledge, our proposed compact formulation has not been proposed before in networking. This destination-based multi-commodity flow formulation is described in the remainder of this section. We then describe our optimization objectives as concave functions in the subsequent sections so that the optimization problems can be solved with convex optimization.

Traffic demand matrix. We denote the *traffic demand* from node ℓ to node k as $T_{k\ell} \geq 0$. Accordingly, we refer to the corresponding $n \times n$ matrix T as the *traffic demand matrix*. As a node k has no traffic to itself that requires transport on the network, we conveniently redefine T_{kk} to be

$$T_{kk} = -\sum_{\ell \neq k} T_{k\ell},$$

the negative of the total traffic demand for, and exiting at, node k. With this definition of T_{kk}, we have

$$\sum_{\ell} T_{k\ell} = 0,$$

i.e., $T\mathbf{1} = 0$, where $\mathbf{1}$ is the vector with all entries equal to one. As defined, T is a Metzler matrix. Note that the traffic matrix T gives us the IE pair traffic (the $n(n-1)$ off-diagonal entries $T_{k\ell}$, $k \neq \ell$) as well as the total traffic demand for each of the n nodes ($-T_{kk}$).

Multi-commodity flow conservation. The traffic flows from ingress node to egress node over a network with m directed edges, as described by its incidence matrix $A \in \mathbf{R}^{n\times m}$, where

$$A_{ij} = \begin{cases} +1 & \text{if edge } j \text{ enters node } i \\ -1 & \text{if edge } j \text{ leaves node } i \\ 0 & \text{otherwise.} \end{cases}$$

We assume that the network is completely connected, *i.e.*, there is a directed path from any node to any other, which is typically the case for backbone networks.

We allow the splitting or aggregation of network flows that are destined to the same egress node. Let $F_{kj} \geq 0$ denote the *flow* on edge j that is destined for destination k. As mentioned, this is a multi-commodity flow problem, with n different flows labeled by their destination or egress node k.

At each node, and for each of the n flows, we must have flow conservation, taking into account the ingress flow at the node, the flow entering the node from incoming edges, the flow leaving the node over outgoing edges, and (when the node is the egress node) the flow egressing from the node. For a node $i \neq k$ (*i.e.*, not the egress node), the ingress flow plus the net flow into the node must sum to zero:

$$T_{ki} + \sum_j A_{ij} F_{kj} = 0, \quad i, k = 1, \ldots, n, \quad i \neq k. \tag{3.1}$$

By net flow, we mean the sum of flows entering on incoming edges minus the sum of the flows leaving on outgoing edges. At the destination node, all the traffic exits, so we have

$$\sum_j A_{ij} F_{ij} = \sum_\ell T_{i\ell} = -T_{ii}, \quad i = 1, \ldots, n. \tag{3.2}$$

The lefthand side is the net flow into node i, and the righthand side is the total of all traffic exiting the network at node i. Equation (3.2) is identical to (3.1) for $k = i$. So (3.1) holds for all $i, k = 1, \ldots, n$. In fact, the n equalities (3.2) hold automatically, which can be seen by summing (3.1) over all edges, so they are redundant. Therefore, we can simply express multi-commodity flow conservation in a compact matrix formula as

$$T + FA^T = 0. \tag{3.3}$$

Edge capacities. The total traffic on edge j is $\sum_k F_{kj}$. In the simplest model, each edge has a capacity that cannot be exceeded, *i.e.*,

$$\sum_k F_{kj} \leq c_j, \quad j = 1, \ldots, m, \tag{3.4}$$

where c_j is the *capacity* of edge j. This can be written as $F^T\mathbf{1} \leq c$, where the inequality is elementwise.

Feasible traffic demands. A traffic demand matrix T (with $T_{k\ell} \geq 0$ for $k \neq \ell$ and $T\mathbf{1} = 0$) can be supported by the network if there exists $F \geq 0$ for which (3.4) and (3.1) hold, *i.e.*,

$$F \geq 0, \qquad T + FA^T = 0, \qquad F^T\mathbf{1} \leq c. \tag{3.5}$$

This set of inequalities, together with $T_{k\ell} \geq 0$ for $k \neq i$ and $T\mathbf{1} = 0$, defines a polyhedron, which we denote as $\mathcal{T}$. We refer to a traffic demand matrix T as feasible if $T \in \mathcal{T}$ (*i.e.*, a feasible traffic demand matrix is one for which there is a set of edge flows that respects flow conservation and edge capacities).

3.3 Formulation of Circuit Allocation Problems

3.3.1 General Approach

To formulate our circuit allocation problems as convex optimization problems, we define a utility function $\phi_{k\ell}(T_{k\ell})$ for each IE pair (k, ℓ), $k \neq \ell$, that computes the *utility* of allocating a circuit with capacity $T_{k\ell}$ to IE pair (k, ℓ) (*i.e.*, a circuit that can support traffic demand up to $T_{k\ell}$). We use the compact notation $\phi(T)$ to denote the matrix with entries $\phi_{k\ell}(T_{k\ell})$ when $k \neq \ell$, and we set the diagonal entries of $\phi(T)$ to one. As discussed below, for both the real-time-based and history-based circuit allocation formulations, $\phi_{k\ell}(T_{k\ell})$ is defined (and required) to be an increasing concave function.

To fairly allocate network resources to implement circuits for different IE pairs, we use the well-known utility fairness notion called α-fairness [57], which is defined as follows:

$$U(f) = \begin{cases} \frac{f^{1-\alpha}}{1-\alpha} & \text{for } \alpha \geq 0 \text{ and } \alpha \neq 1 \\ \log f & \text{for } \alpha = 1 \end{cases} \tag{3.6}$$

Depending on the choice of α, different notions of fairness can be achieved. For example, maximum utility is obtained when $\alpha = 0$, proportional fairness is obtained when $\alpha \to 1$, and max-min fairness is obtained when $\alpha \to \infty$. In practice, a large α is sufficient to ensure max-min fairness. For any $\alpha \geq 0$, $U(f)$ is an increasing concave function. We then formulate the circuit allocation problem as follows:

$$\begin{array}{ll} \text{maximize} & \sum_{k,\ell} U(\phi_{k\ell}(T_{k\ell})) \\ \text{subject to} & T \in \mathcal{T}, \end{array}$$

where $T \in \mathcal{T}$ corresponds to the set of constraints defined in (3.5). We refer to the objective as the *total network utility.* Since an increasing concave function of a concave function is still concave [16], $U(\phi_{k\ell}(T_{k\ell}))$ is a concave function of $T_{k\ell}$. The objective is a sum of concave functions, and therefore it is also a concave function. Maximizing a concave function subject to convex constraints (*i.e.*, linear equality and inequality constraints) is a convex optimization problem.

Since the objective is an increasing function of T, we see that at the optimal point, all edge traffic will actually be equal to the edge capacity (*i.e.*, we will have $F^T\mathbf{1} = c$). Therefore, we can replace the inequality $F^T\mathbf{1} \leq c$ in (3.5) with the equality constraint

$F^T\mathbf{1} = c$. The convex optimization problem then becomes

$$\begin{array}{ll} \text{maximize} & \sum_{k,\ell} U(\phi_{k\ell}(T_{k\ell})) \\ \text{subject to} & F \geq 0, \\ & T + FA^T = 0, \\ & F^T\mathbf{1} = c. \end{array} \tag{3.7}$$

with variables T (the traffic demands that can be supported) and F (the detailed network flows).

In the remainder of this section, we consider two versions of the circuit allocation problem. In the first case, we consider the deployment setting in which *real-time traffic measurements* are possible, and we can dynamically allocate circuits on a frequent basis in response to changing traffic. In the second case, we consider the deployment setting in which we allocate circuits based on *historical traffic patterns*. In both versions of the problem, we optimize for *utility max-min fairness* by using a sufficiently large α value in (3.6). The two problems differ in how we define the utility functions $\phi_{k\ell}(T_{k\ell})$ for the IE pairs.

3.3.2 Real-Time-Based Allocation

In this section, we consider the deployment setting in which actual traffic can be measured in real-time at a reasonable timescale, and that circuits can be dynamically reconfigured. In particular, let $r_{k\ell}$ be the traffic rate at the current measurement interval for IE pair (k, ℓ). Intuitively, the traffic pattern for the next time interval should be similar to the current measurement interval if the measurement/reconfiguration interval is sufficiently short. Therefore, we aim to allocate circuit capacities proportional to the current traffic rates, but we want to fully allocate all network resources even when circuit allocations cannot be further increased for some IE pairs. In particular, we define

$$\phi_{k\ell}(T_{k\ell}) = \frac{T_{k\ell}}{r_{k\ell}} \tag{3.8}$$

By defining the utility function $\phi_{k\ell}(T_{k\ell})$ this way, the solution to network optimization problem (3.7) corresponds to the *weighted max-min fair* solution.

3.3.3 History-Based Allocation

Alternatively, in this section, we consider the deployment setting in which real-time traffic measurements are not possible. In this case, we make use of historical traffic statistics to predict the traffic demands for a given time period. Previous studies have shown that the aggregate traffic at the core of the network tends to be very smooth and that it follows strong diurnal patterns. In particular, historical traffic demands during a particular time of day (*e.g.*, 11:00-11:30am on a weekday) are a good indicator of expected future traffic demands over the same time of day. Let $\mathbf{r}_{k\ell} = \{r_{k\ell}(1), r_{k\ell}(2), \ldots, r_{k\ell}(t)\}$ be a collection of t historical traffic measurements taken at a particular time of day for the IE pair (k, ℓ). The corresponding empirical cumulative distribution function (CDF) $\Phi_{k\ell} : \mathbf{R}_+ \to [0, 1]$ maps a circuit capacity $T_{k\ell}$ (*i.e.*, the amount of traffic demand that the circuit can support) to the fraction of $\mathbf{r}_{k\ell}$ data points that can be supported:

$$\begin{aligned} \Phi_{k\ell}(T_{k\ell}) &= \frac{\#measurements \leq T_{k\ell}}{t} \\ &= \frac{1}{t}\sum_{i=1}^{t} I[r_{k\ell}(i) \leq T_{k\ell}] \end{aligned} \tag{3.9}$$

where $I[r_{k\ell}(i) \leq T_{k\ell}]$ is the indicator that the measurement $r_{k\ell}(i)$ is less than or equal to the circuit capacity $T_{k\ell}$.

From an empirical CDF $\Phi_{k\ell}(T_{k\ell})$, we derive an increasing concave function $\phi_{k\ell}(T_{k\ell})$

by curve fitting the empirical CDF. That is, for each data point $r_{k\ell}(i) \in \mathbf{r}_{k\ell}$, we have the corresponding empirical CDF data point $\Phi_{k\ell}(r_{k\ell}(i))$. In general, $\Phi_{k\ell}(T_{k\ell})$ is not concave. However, for traffic values above the median measured data rate, the corresponding probability density function (PDF) of traffic is typically decreasing, which is reasonable to assume. Therefore, we simply curve fit $\phi_{k\ell}(T_{k\ell})$ to all the empirical CDF data points above the median data rate in $\mathbf{r}_{k\ell}$ (*i.e.*, for all $\Phi_{k\ell}(r_{k\ell}(i))$ such that $r_{k\ell}(i) \geq \text{median}(\mathbf{r}_{k\ell})$) using an increasing concave functional form. In general, any increasing concave function can be used as the parametric form for curve fitting. As we shall see in §3.4.2, we have found that an increasing concave *piecewise linear* (PWL) functional form can accurately approximate the empirical CDFs above the corresponding median historical data rate. As another example, fitting the historical data rates to a log-concave functional form would be another natural way to accurately approximate the empirical CDFs.

By deriving the utility function $\phi_{k\ell}(T_{k\ell})$ from the empirical CDF of the historical traffic, we are maximizing the *probability* that the allocated circuits can handle future traffic demands if future traffic demands follow similar traffic patterns as the measured historical traffic. Correspondingly, the solution to the network optimization problem (3.7) corresponds to the *utility max-min fair* solution where the utility function is derived from historical traffic.

3.3.4 Deriving Per-IE Pair Circuit Configurations

By solving for the convex optimization problem (3.7) with the utility functions defined in either §3.3.2 or §3.3.3, we know what circuit capacities $T_{k\ell}$ can be realized for each IE pair (k, ℓ). However, in our destination-based multi-commodity flow formulation, a *flow* corresponds to all traffic that are destined for the same destination k, and the flow assignment variables $F_{kj} \geq 0$ denote the flow on edge j that is destined for destination k. As discussed earlier, this formulation enables us to reduce the number of variables by a factor of $n-1$, which allows us to scale our approach to far larger networks.

To derive the actual circuit configurations on a per-IE pair basis, we have to disaggregate a single destination flow into parts associated with the different IE pairs. This has nothing to do with the optimization method, and does not affect what traffic profiles that we are able to support.

As observed earlier, the solution must satisfy $F^T\mathbf{1} = c$ (*i.e.*, all the edge capacity is used). Given this constraint, we can show that for each flow with a given destination, there are no (directed) cycles. To see this, suppose that for destination k there is a nonzero (*i.e.*, positive) directed cycle. This means there are edges $e_1, \ldots, e_p$ that form a directed cycle, and the flow destined for node k is positive on each of these edges. This implies that we can reduce the flow destined for node k on each of these edges by some positive amount, and remain feasible. By reducing the flow on each of these edges, we now have unused capacity on these edges, which we can use by assigning (for example) to the IE pairs associated with those edges. This increases these IE pair traffic values, which increases the objective, which shows the original flow was not optimal. Therefore, the optimal solution contains no (directed) cycles for each destination flow. We can exploit this property in deriving the per-IE pair circuit configurations.

In particular, we start with the traffic matrix F, which gives the flow on each edge for each destination. Our goal is to give a more detailed flow description $Z_{k\ell,j} \geq 0$, which is the flow on edge j for the IE pair (k, ℓ). For each IE pair (k, ℓ), the edges with nonzero $Z_{k\ell,j}$ values show us the route or routes that IE pair (k, ℓ) takes. This must satisfy the obvious flow conservation, where it is conserved for all nodes other than k or ℓ, the traffic $T_{k\ell}$ enters at node ℓ and leaves at node k. These detailed flows must satisfy $Z_{k\ell,j} \geq 0$ and

$$\sum_{\ell} Z_{k\ell,j} = F_{kj},$$

but otherwise are completely arbitrary. We describe two simple methods for constructing Z given F, but many other methods could be used as well.

Indeed, any method that attributes flow to each IE pair (k, ℓ) such that the remaining flow satisfies all the conditions described above (though with $T_{k\ell}$ set to zero) will work. The lack of flow cycles ensures that all flow can be attributed to IE pairs.

Greedy assignments. Consider an IE pair (k, ℓ). We can route the traffic from node ℓ to node k in a greedy way. Starting at node ℓ, route all the flow along an outgoing edge j with $F_{kj} \geq T_{k\ell}$, if there is such an edge. If there is no such edge, we will need to split the flow into two or more edges. We repeat this until we get to the destination. Then we subtract these flows from the F matrix, which leaves the flows destined for node k, other than the flow originating at node ℓ. We repeat the process. This method always works; it tends to avoid splitting flows.

Proportional assignments. Alternatively, we can route the traffic for IE pair (k, ℓ) from node ℓ by splitting the flow proportionally across outgoing edges j_h to a node h that have $F_{kj_h} > 0$. Our multi-commodity flow formulation ensures that

$$\sum_h F_{kj_h} \geq T_{k\ell}.$$

In particular, we assign to the detailed flow

$$Z_{k\ell,j_h} = T_{k\ell} \left(\frac{F_{kj_h}}{\sum_h F_{kj_h}} \right), \quad \text{for each } j_h. \tag{3.10}$$

We repeat this by proportionally splitting each $Z_{k\ell,j_h}$ across the outgoing edges of node h until we get to the destination. Like the greedy assignment method, we subtract these detailed flows from the F matrix. We repeat this process for other IE pairs. This method also always works. It tends to split the flows a lot; more specifically, whenever a flow splits at a node, then all IE pairs will also split there.

3.4 Evaluation Setup

3.4.1 Network and Traffic Matrices

We have evaluated our proposed network optimization framework on the optimal circuit allocation problem on a real, large PoP (point of presence)-level backbone network, namely the Abilene network [40]. The Abilene network has been studied and discussed in the research literature. Its network topology, traffic dataset, and routing information are available in the public domain [90]. In particular, Abilene is a public academic network in the US with 11 nodes interconnected by OC192, 9.92 Gbits/s links. (Abilene actually has another secondary core router at Atlanta, but it only connects to the primary Atlanta core router and has much less traffic. To simplify the topology, we merged this secondary core router into the primary Atlanta core router, including all of its traffic.)

To evaluate the Abilene network, we use real traffic matrices that have been collected by a third party [90] in our simulations. We also use these traffic matrices in our experiments to derive the circuit configurations using our proposed network optimization algorithms. A traffic matrix consists of the requested traffic rates for every source-destination pair within a 5-minute interval. Therefore, these traffic matrices provide a snapshot of real total demand offerings between each IE pair in the Abilene network every five minutes. The traffic matrices are derived based on the flow information collected from key locations of a network by traffic monitors, such as Netflow [1]. Then the flow information is transformed into the demand rate of each IE pair in a traffic matrix based on the routing information in the network. We collected the traffic matrices in each network for an extended period of time to represent the historical traffic measurements and simulation traffic load. The detail information of the traffic matrices used is summarized in Table 3.1.

In particular, for *history-based* circuit allocation, as described in §3.3.3, we use the historical traffic patterns during a particular time of day (3:00-3:30pm on a Wednesday)

Table 3.1: Traffic data for Abilene.

Network	Collection Period	Historical Traffic for Allocation	Test Traffic for Evaluation	Time Interval
Abilene	05/01/04 to 07/02/04	05/01/04 to 06/18/04	06/19/04 to 07/02/04	5 min

over a 7 weeks period from 05/01/2004 to 06/18/2004. Since the dataset offers the traffic matrices at 5-minute intervals, each IE pair has 42 historical traffic data points across the analyzed period. For our evaluations, we simulated the network traffic at the same time of day (3:00-3:30pm on a Wednesday) in the following two weeks from 06/19/2004 to 07/02/2004. This gives another 12 traffic matrices for evaluation.

For *real-time-based* circuit allocation, as described in §3.3.2, we also use the 12 traffic matrices during the two weeks from 06/19/2004 to 07/02/2004 for evaluation. For each of the 12 test traffic matrices, we interpolate the test traffic matrix with the test traffic matrix from the previous 5-minute interval, and we use this interpolated traffic matrix to define the current measured traffic rate $r_{k\ell}$ in the utility function $\phi_{k\ell}(T_{k\ell})$ shown in (3.8).

3.4.2 Modeling Traffic Statistics

As discussed in §3.3.3, for each IE pair, we model the distribution of historical traffic patterns by an empirical cumulative distribution function (CDF). In particular, for each IE pair, we use the collection of historical traffic data points $\mathbf{r}_{k\ell} = \{r_{k\ell}(1), r_{k\ell}(2), \ldots, r_{k\ell}(t)\}$ to define a corresponding empirical CDF $\Phi_{k\ell} : \mathbf{R}_+ \to [0,1]$, as shown in (3.9), and we use curve fitting to fit the empirical CDF data points $\Phi_{k\ell}(r_{k\ell}(i))$ to derive an increasing concave approximation function $\phi_{k\ell}(T_{k\ell})$. As noted in §3.3.3, the CDF of a historical traffic distribution should be concave above the median traffic level. This is because the probability density function (PDF) of traffic should be decreasing above the median level.

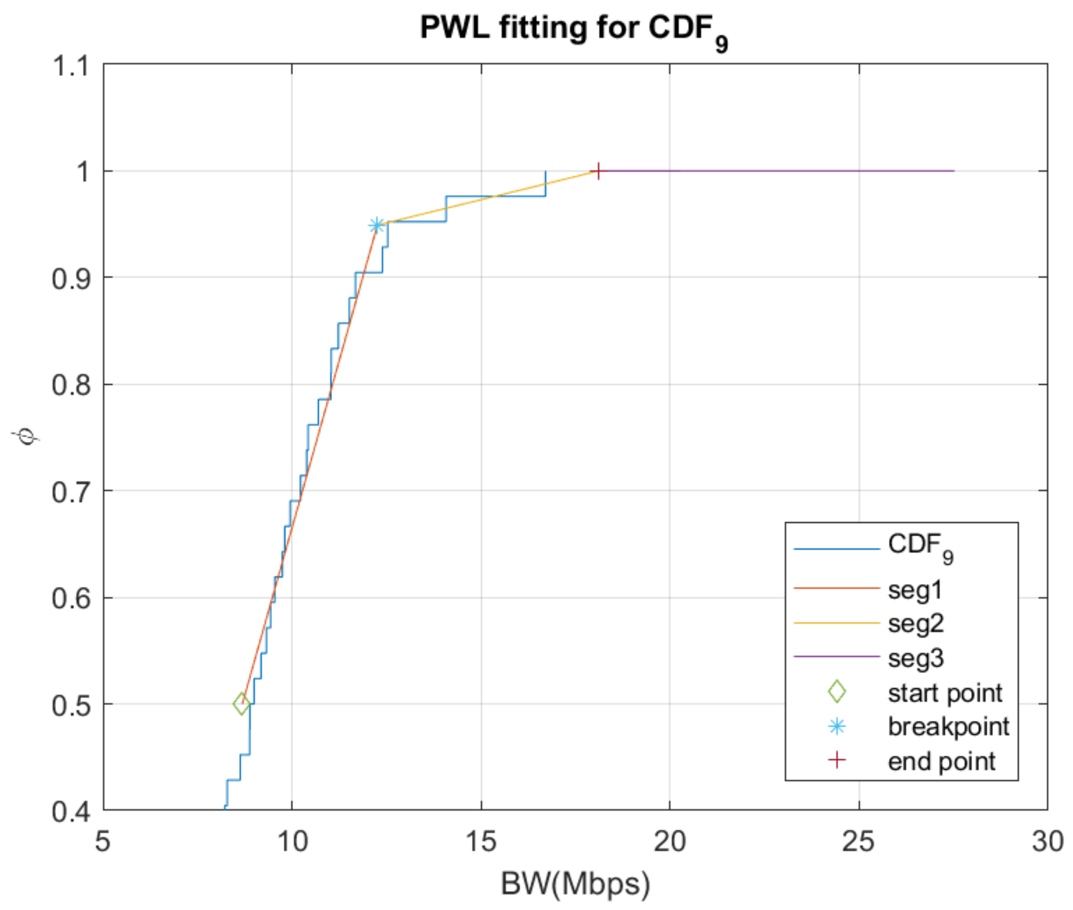


Figure 3.2: An example PWL curve fitting of the historical traffic CDF for the flow from Atlanta to Seattle.

Therefore, we can accurately approximate the empirical CDF as a concave function by curve fitting to those empirical CDF data points at or above the median data point for all IE pairs.

In our evaluations, we use a piecewise linear (PWL) curve fitting to approximate the empirical CDF. Fig. 3.2 shows an example a PWL curve fitting for the IE pair traffic from Atlanta to Seattle. In particular, the PWL curve shown corresponds to the empirical CDF of the 42 data points collected over the 7 weeks period between 05/01/2004 and 06/18/2004. The PWL curve is fitted to all data points at or above the median level. In our experiments, we used CVXPY to implement a piecewise-linear curve fitting approach based on least-square fitting to a fixed number of segments (*e.g.*, 3 segments). More sophisticated piecewise linear curve fitting approaches (*e.g.*, [54]) can be used as well.

3.4.3 Circuit Allocation

We performed the circuit allocation for all 11 cities in the Abilene network, corresponding to 110 IE pairs, by solving the convex optimization problem (e-network-opt) in §3.3. For real-time-based circuit allocation, we use $\phi(T)$ as defined in §3.3.2, and for history-based circuit allocation, we use the PWL curve fitting approach shown above to derive $\phi(T)$, as discussed in §3.3.3. For α-fairness (*cf.* (3.6)), we assume $\alpha = 2$. To solve the convex optimization, we use CVXPY [32] with MOSEK [7].

3.4.4 Re-Routing over Circuits for Adaptation

Although both our real-time-based and history-based circuit allocation algorithms aim to allocate circuit capacities so that actual traffic can be carried directly by the allocated circuits, traffic fluctuations or unexpected traffic changes can lead to inadequate capacities along direct circuits. One way to handle the excess traffic is to adaptively re-route the excess traffic over circuits with spare capacity. Since our circuit allocation algorithms are

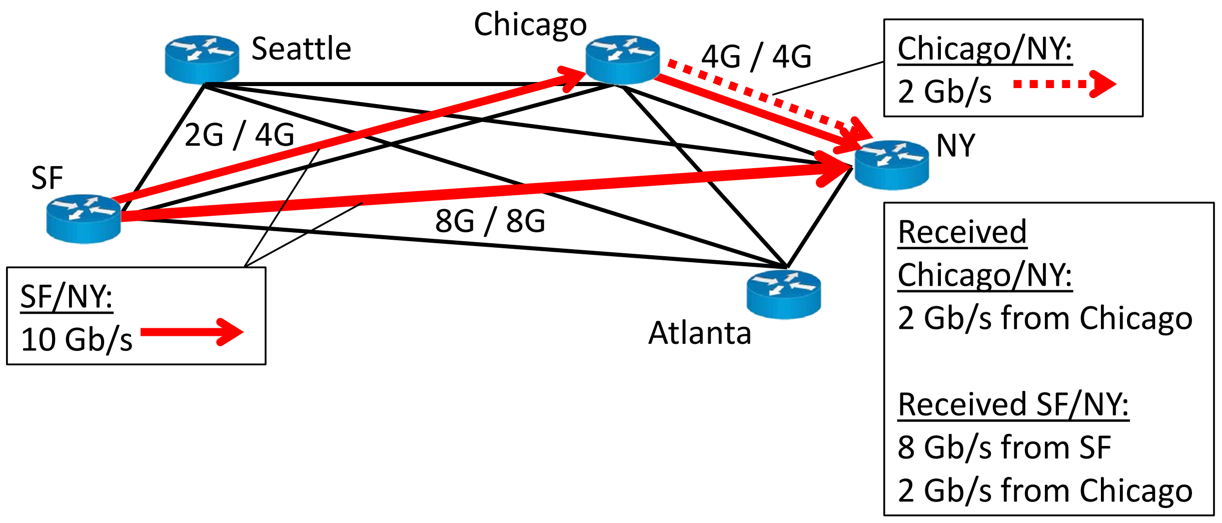


Figure 3.3: When excess traffic occurs from SF to NY, we can re-route it using the residual circuit capacity of the path through for example Chicago.

designed to create direct circuits between every IE-pairs, the logical network topology becomes a fully-connected mesh.

Consider the example depicted in Fig. 3.3. Suppose the circuit capacity from SF (San Francisco) to NY (New York) is 8 Gb/s, and suppose the circuit capacities from SF to Chicago and Chicago to NY are both 4 Gb/s. Normally, we expect a circuit to have enough capacity for its direct traffic. For example, in Fig. 3.3, given 2 Gb/s of traffic from Chicago to NY, all of its traffic can be directly sent through the network using the circuit from Chicago to NY. However, suppose we have a 10 Gb/s burst of traffic from SF to NY, then there would be 2 Gb/s of excess traffic because the circuit capacity from SF to NY is only 8 Gb/s. When this occurs, an adaptive re-routing mechanism can be triggered to re-route the 2 Gb/s of excess traffic over alternative circuit routes, for example through Chicago by the utilizing the residual circuit capacity available along SF-Chicago and Chicago-NY.

As can be seen from this example, with the help of adaptive re-routing, we can increase network throughput without the need to create new circuits on-the-fly. Although this adaptive re-routing does rely on electronic routing at intermediate nodes, it is only used as a secondary mechanism to handle excess traffic. The majority of traffic is still expected to be carried by the corresponding direct circuits. Therefore, the route processing portion of a unified circuit/packet switch can remain simple.

In our experiments, we consider two re-routing methods. One is based on a variant of the well-known backpressure-based re-routing algorithm [85] that guarantees optimal re-routing. In the modified version of the backpressure-based re-routing algorithm, a unified switch maintains a separate queue for packets for each destination, and it transmits packets on the direct circuit as long as the circuit has sufficient capacity. Insufficient capacity is detected when the queue of packets for the direct circuit exceeds some threshold L_{max}. When this occurs, packets are re-routed using the backpressure algorithm. The re-routing is optimal in the sense that if a traffic pattern can be handled by re-routing over the logical fully-meshed network of circuits, then the re-routing algorithm is guaranteed to succeed in re-routing all traffic to their destinations. In §3.5.2, we refer to this re-routing approach as "OptRR" for optimal re-routing.

Alternatively, we also consider a simple greedy re-routing algorithm that simply re-routes the excess traffic over the outgoing circuit with the most residual capacity. Suppose $T_{m\ell}$ is the circuit capacity allocated to the circuit from the current node ℓ to node m, and suppose $\mu_{m\ell}$ is the measured rate of traffic sent on the circuit from the current node ℓ to node m in the current measurement interval. Then the amount of "residual capacity" on the circuit from the current node ℓ to m is simply $T_{m\ell} - \mu_{m\ell}$. A simple greedy algorithm is just to re-route traffic to node m via the circuit to node m with the most residual capacity rather than directly to destination k. This greedy approach only requires information that can be measured locally, but it is not optimal. We include this re-routing method in our experiments to show that even this simple approach is effective with our circuit allocation methods. In §3.5.2, we refer to this re-routing approach as "GreedyRR" for greedy re-routing.

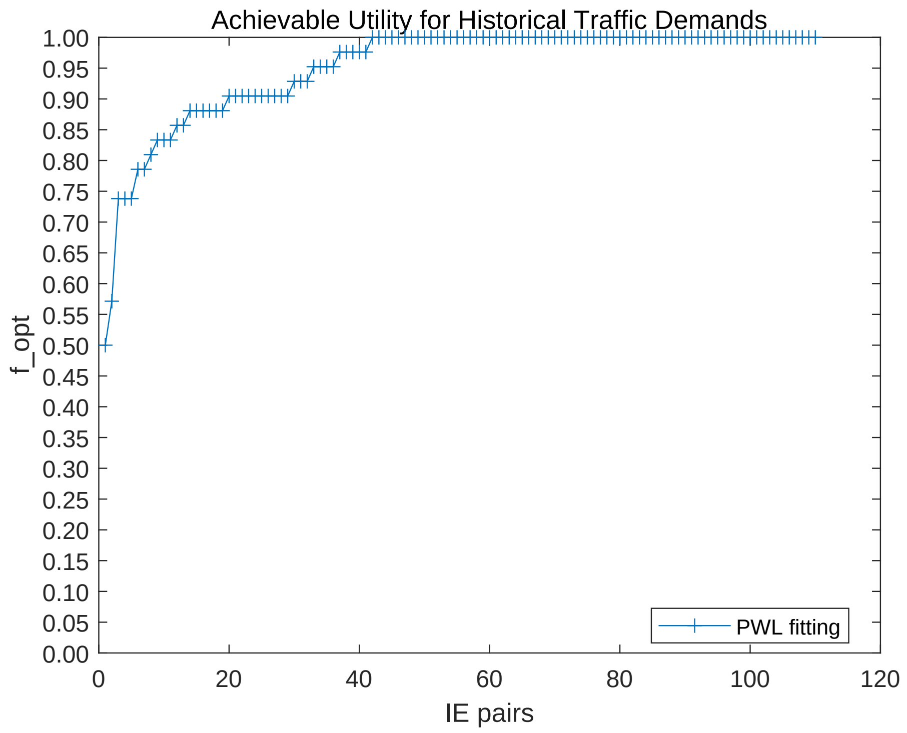
3.5 Experimental Results

In this section, we first evaluate the performance of our history-based circuit allocation algorithm in terms of what fraction of the historical traffic patterns that the allocated circuits can handle as well as the fraction of test traffic patterns that the allocated circuits can handle. We then compare the performance of circuit-switching approaches using our circuit allocation methods, namely the real-time-based and history-based circuit allocation approaches, with a conventional packet-routing algorithm, OSPF [60] in §3.5.2. We extend our circuit-switching approaches with adaptive re-routing in cases when the circuit capacity is not enough, as discussed in §3.4.4. This re-routing approach is also evaluated in §3.5.2. Our evaluations show that our circuit allocation algorithms can indeed accommodate most of the actual traffic, and adaptive re-routing over the allocated circuits can effectively accommodate excess traffic even under heavy traffic loads.

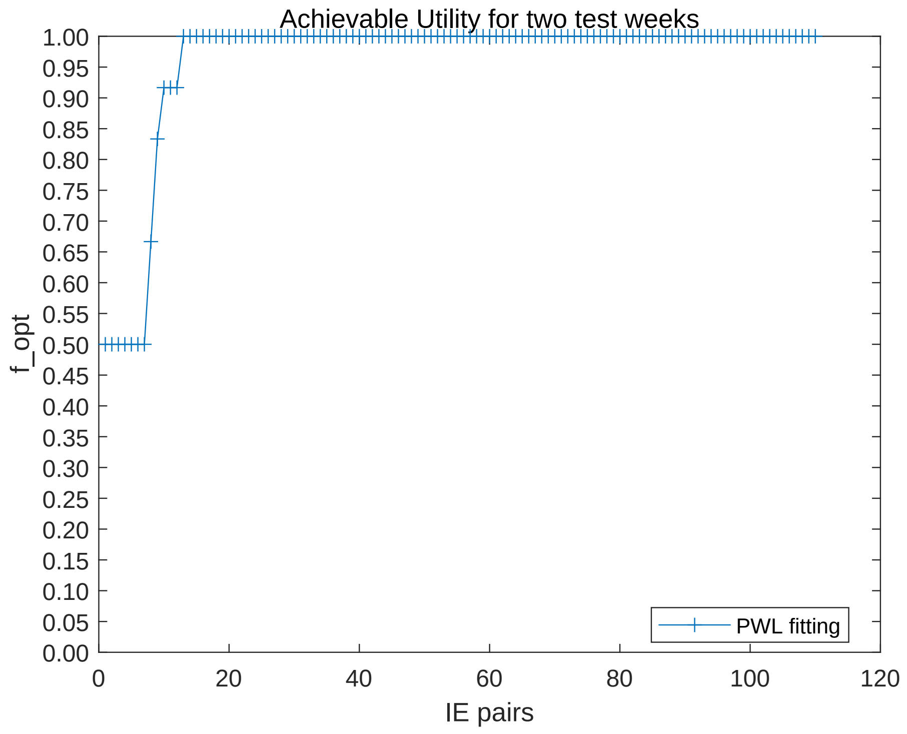
3.5.1 Evaluation of History-Based Circuit Allocation

Figure 3.4 shows the fractions of the data points whose traffic demands may be accommodated by the optical circuit allocation solved by the convex optimization problem solver for all 110 IE pairs. The achievable f_{opt} on the Y-axis means the allocation $T_{k\ell}$ is no less than the fraction f_{opt} of the traffic data points for IE pair (k, ℓ). For example, $f_{opt} = 0.9$ for an IE pair means $T_{k\ell}$ is greater or equal to 90% of the traffic data points for that IE pair (k, ℓ) at a given time, and $f_{opt} = 1$ means $T_{k\ell}$ can accommodate all of the traffic data points for that IE pair. In particular, figure 3.4(a) shows the coverage of the historical traffic patterns, and figure 3.4(b) shows the coverage of the test traffic patterns.

As can be seen from figure 3.4(a), the circuit configuration is able to accommodate all historical traffic data points for more than two thirds of all 110 IE pairs. The smallest fraction occurs at 0.5, and that is for only one flow. When the circuit configuration is



(a) Historical traffic.



(b) Test traffic.

Figure 3.4: Achievable utility for historical and test traffic demands

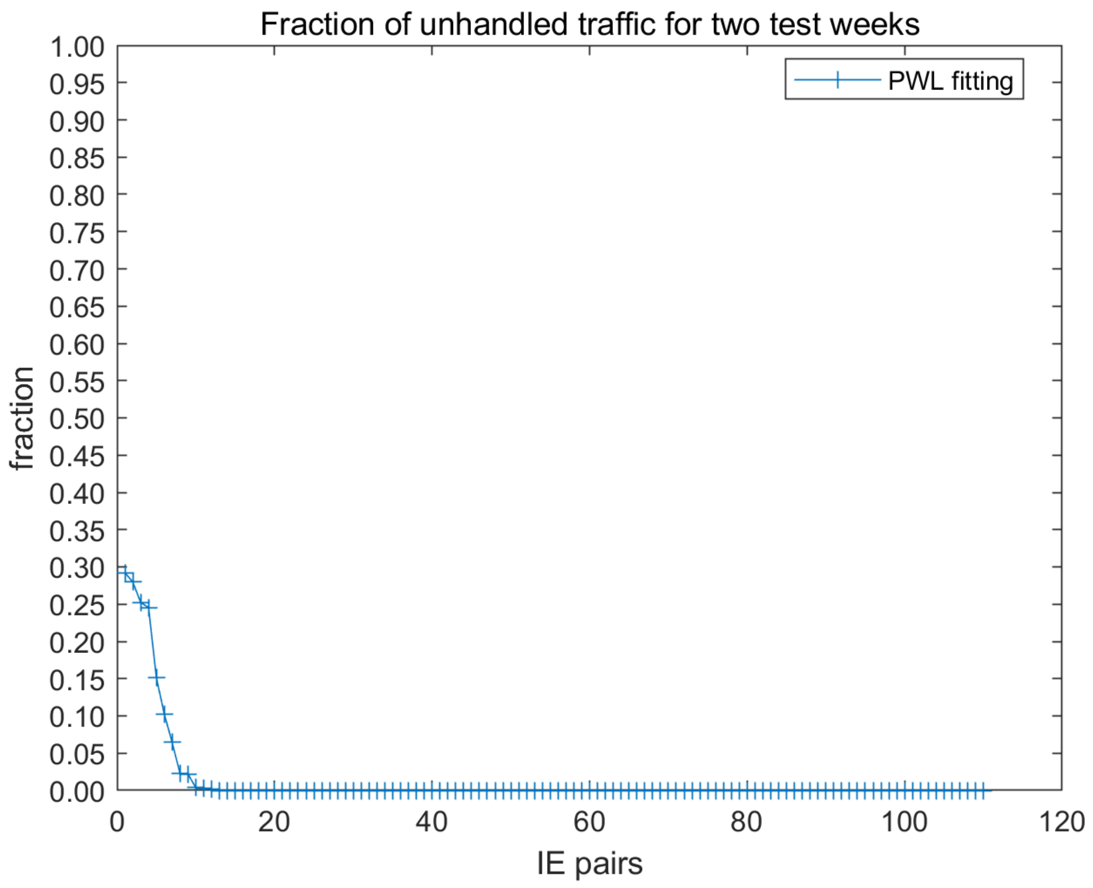


Figure 3.5: Fraction of unhandled traffic for two test weeks

applied to the two test weeks, figure 3.4(b) shows that this configuration can accommodate all data points for more than 80% IE pairs.

Figure 3.4 only considers the traffic demands that are strictly lower than the optical circuit bandwidth. If a traffic demand is slightly higher than the given circuit bandwidth, the circuit allocation is considered to fail to accommodate that data point. As can be seen from figure 3.4(b), the circuit allocation of some IE pairs fails to accommodate half of the traffic demand data points. However, the actual unhandled traffic in this case may be small. Therefore, figure 3.5 is used to show the amount of unhandled traffic for the test weeks when our circuit allocation is used.

As can be seen from figure 3.5, our history-based circuit allocation can accommodate all traffic demands for 90% IE pairs in the test weeks, and only less than 30% traffic from the worst-case IE pair is unhandled by the allocated bandwidth.

3.5.2 Performance Evaluations

To evaluate the performance of our circuit allocation approach, we compare the following:

- *OSPF*: This is conventional packet routing over the Abilene network in which the routing paths are determined using the Open Shortest Path First (OSPF) protocol [60], which is used for packet routing. The routing paths are based on Dijkstra's single shortest path algorithm. This conventional approach serves as a baseline for our evaluations.

- *RT*: The plots labeled "RT" correspond to our real-time-based circuit allocation algorithm for optical circuit-switching. In particular, we consider three cases. The first case corresponds to circuit-switching without re-routing. Here, traffic is also simply sent directly over a fully-meshed network in one logical hop, whose circuit capacities are determined by the algorithm described in §3.3.2. This is labeled as "RT-NoRR." The other two cases correspond to the two methods of re-routing, as discussed in §3.4.4. "RT-GreedyRR" corresponds to greedy re-routing based on residual capacity, and "RT-OptRR" corresponds to optimal re-routing based on a modified version of the backpressure algorithm for re-routing [85].

- *HIST*: The plots labeled "HIST" correspond to our history-based circuit allocation for optical circuit-switching. "HIST-NoRR," "HIST-GreedyRR," and "HIST-OptRR" correspond to the cases of no re-routing, greedy re-routing based on residual capacity, and optimal re-routing based on the backpressure algorithm.

For each method, we simulated the network traffic during the two weeks from 06/19/2004 to 07/02/2004 at the same time of day (3:00-3:30pm on a Wednesday). This provides 12 traffic matrices for evaluation. We measured the drop rates, router hops, router loads, and the percentage of packets being routed. The results presented are averaged over the 12 test cases. To demonstrate the performance of our algorithms under high traffic loads, we normalized the traffic by scaling up the traffic loads until OSPF routing begins to drop packets. That is, a normalized traffic load of 1.0 is the intensity of traffic that causes the network to saturate when conventional packet switching with OSPF is used. To test the robustness of our circuit allocation approaches, we further amplify the traffic intensity beyond this saturation point to see how much more traffic our circuit allocation approaches with re-routing can handle.

Figure 3.6 compares drop rates among real-time-based allocation (RT-NoRR, RT-GreedyRR, and RT-OptRR), history-based allocation (HIST-NoRR, HIST-GreedyRR, and HIST-OptRR), and OSPF. The suffixes NoRR, GreedyRR, and OptRR correspond to no re-routing, greedy re-routing, and optimal re-routing, respectively, as discussed in §3.4.4. The X-axis represents traffic loads which are normalized to the load where OSPF begins to drop packets. As we can see, compared with OSPF, our "re-routing"

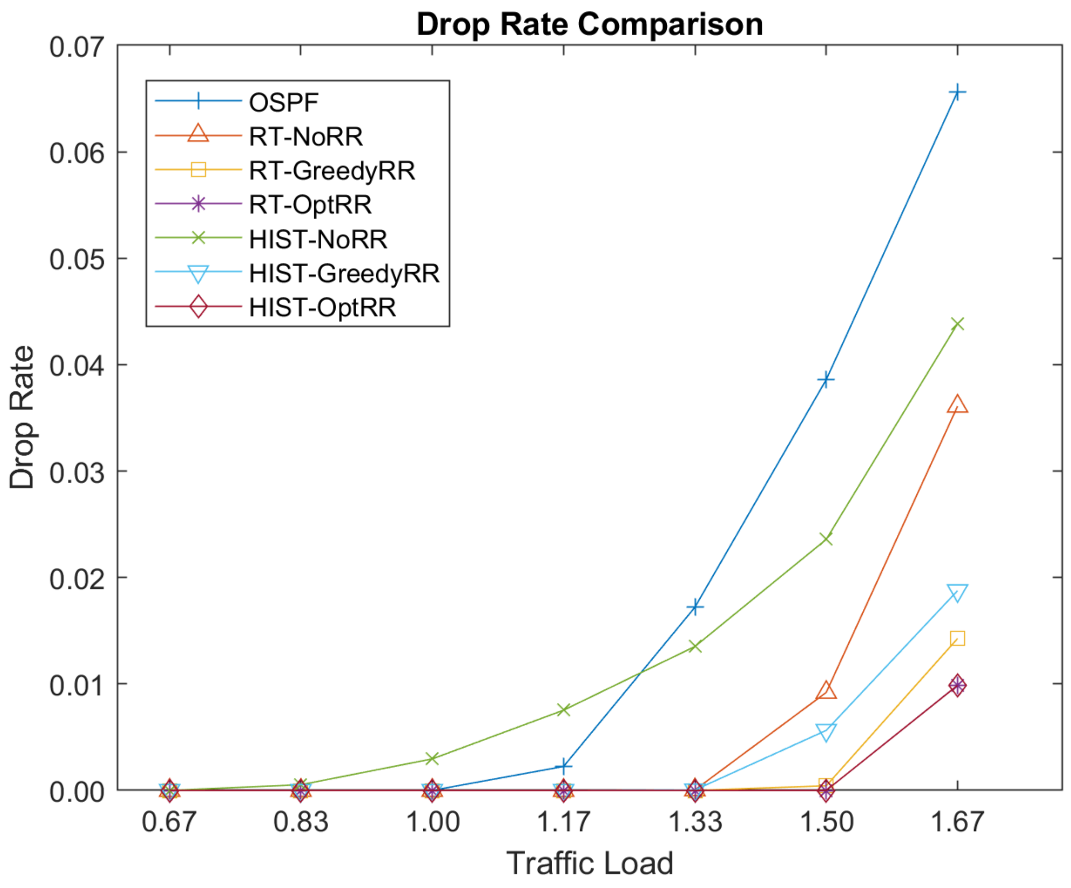


Figure 3.6: Drop rate comparison.

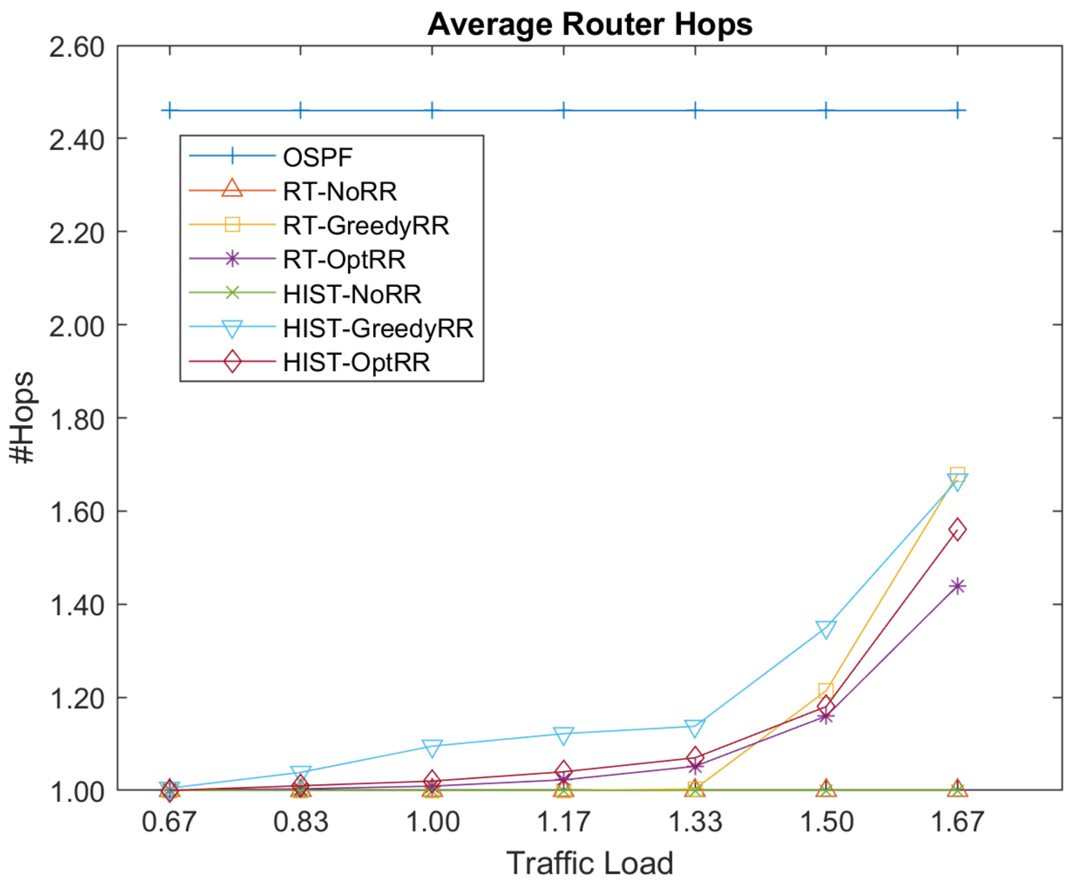


Figure 3.7: Average router hops.

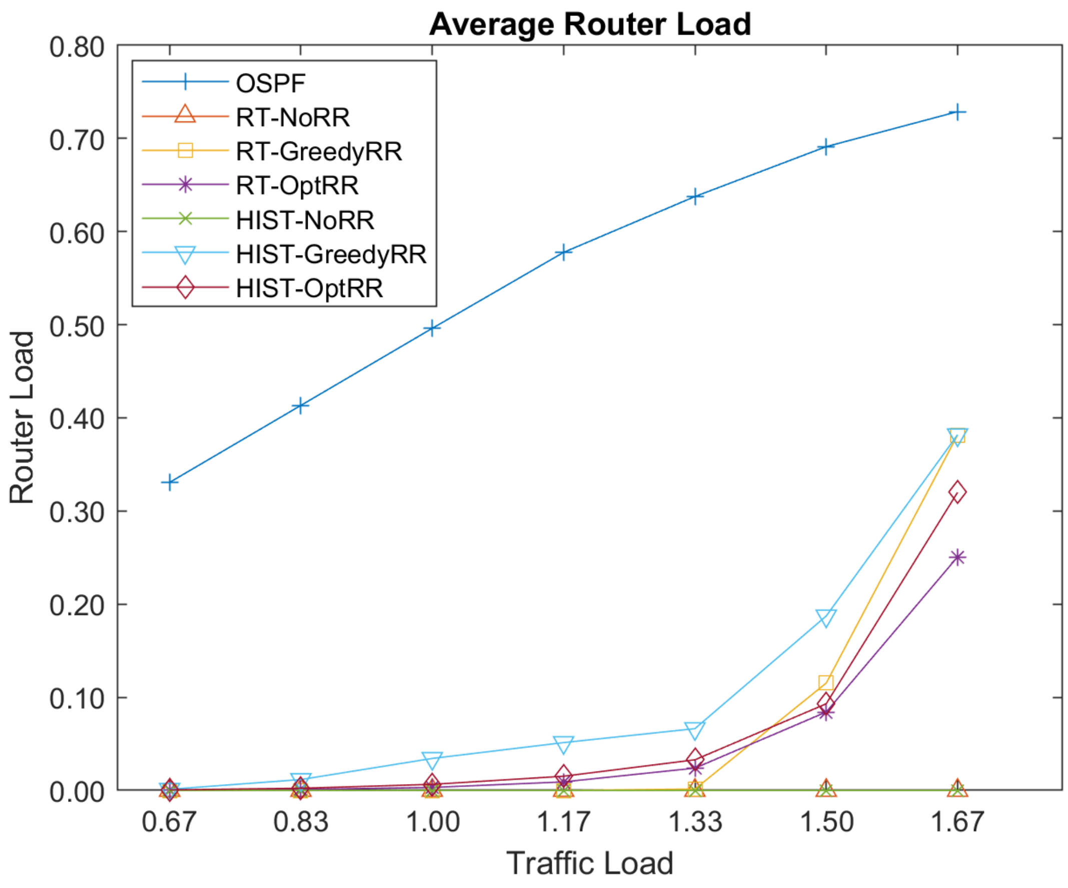


Figure 3.8: Average router load.

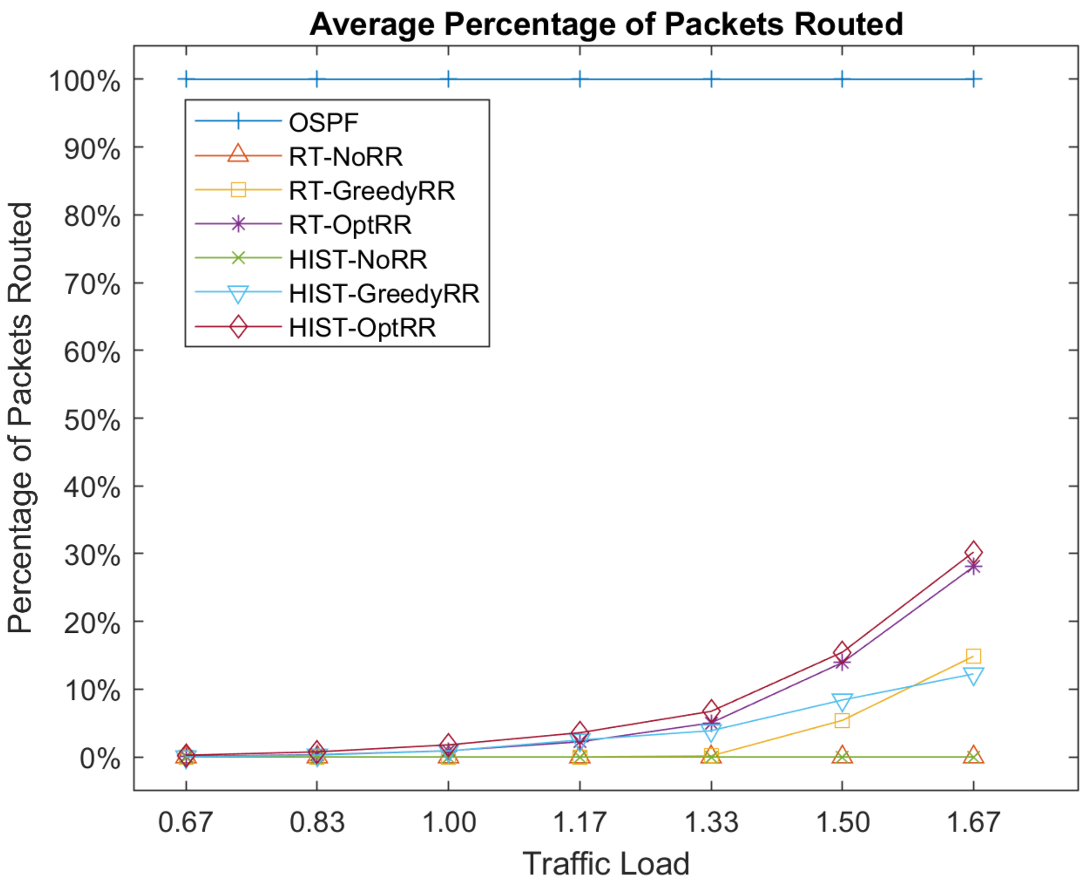


Figure 3.9: Average percentage of packets routed.

approaches (RT-GreedyRR, RT-OptRR and HIST-OptRR) can handle 50% more traffic without dropping packets. Even with greedy re-routing, our history-based approach (HIST-GreedyRR) can handle 33% more traffic without dropping packets. With "no re-routing," our history-based circuit allocation approach (HIST-NoRR) has a negligible drop rate (0.00296) at the normalized load of 1.0 while OSPF has none, but HIST-NoRR has a lower drop rate as traffic continues to scale. As expected, with "no re-routing," our real-time-based approach (RT-NoRR) achieves significantly better results than the history-based approach (HIST-NoRR) because the circuit allocations are based on real-time traffic measurements. Even without re-routing, our real-time-based approach (RT-NoRR) can handle 33% more traffic than OSPF without dropping packets. The reason for the higher throughput is because OSPF always route along the shortest path, whereas our circuits can be configured across multiple paths, optimized to real-time traffic measurements. Although the real-time-based circuit allocation approach performs better, both the real-time-based and the history-based approaches are important, depending on whether or not the deployment scenario allows for real-time measurements and frequent updates or not. The Internet, as implemented today, does not have real-time measurements or allow for frequent dynamic updates, but emerging software defined networking scenarios would provide for that. Our optimization framework supports both settings.

Figure 3.7 shows the number of router hops a packet needs to go through until it reaches its destination. With no re-routing, the real-time-based (RT-NoRR) and history-based (HIST-NoRR) approaches are both always 1 hop over the direct optical circuit, whereas OSPF routing averages 2.46 hops independent of load. With optimal re-routing, both real-time-based (RT-OptRR) and history-based (HIST-OptRR) approaches require very little re-routing for loads up to 1.5X. With greedy re-routing, the real-time-based approach (RT-GreedyRR) also requires very little re-routing for loads up to 1.5X, whereas the history-based approach (HIST-GreedyRR) requires very little re-routing for loads up to

1.33X. As expected, these results show that the real-time-based approach is more accurate than the history-based approach, but better re-routing can compensate for the difference. At higher normalized loads, less re-routing is required when real-time-based allocation is used together with optimal re-routing.

Figure 3.8 shows the average router load among real-time-based allocation (RT-NoRR, RT-GreedyRR, and RT-OptRR), history-based allocation (HIST-NoRR, HIST-GreedyRR, and HIST-OptRR), and OSPF. With "no re-routing" (RT-NoRR and HIST-NoRR), all packets go over direct optical circuits and therefore these approaches have 0 electronic router load. For OSPF, all packets are handled by electronic routers, so as expected, the router load increases with traffic load. When optimal re-routing is employed with both the real-time-based (RT-OptRR) and history-based (HIST-OptRR) approaches, we see that most packets go over direct optical circuits until 1.33X traffic load; after that, the electronic router load increases as more packets get re-routed. As with the average number of hops, we see that the history-based approach (HIST-OptRR) requires a higher router load than the real-time-based approach (RT-OptRR) when the normalized traffic load is increased to 1.67X. This is mostly due to the fact that when packets are re-routed, they go through a high number of intermediate nodes (a higher number of hops). When greedy re-routing is employed, router loads are comparable at 1.67X normalized loads for both real-time-based (RT-GreedyRR) and history-based (HIST-GreedyRR) allocation approaches.

Finally, figure 3.9 shows the average percentage of packets that require routing. For OSPF, 100% of the packets are routed. With "no re-routing" (RT-NoRR and HIST-NoRR), none of the packets are routed since they all go over direct optical circuits. With "re-routing" (RT-GreedyRR, RT-OptRR, HIST-GreedyRR, and HIST-OptRR), most packets go over direct optical circuits until 1.33X traffic load (less than 10% of packet gets re-routed at this load); after that, an increasing percentage of packets get re-routed. When packets have to re-routed, the optimal re-routing approaches (RT-OptRR and HIST-OptRR) route a higher percentage of packets, but most of the time by a fewer a number of hops in comparison with greedy re-routing (RT-GreedyRR and HIST-GreedyRR).

3.6 Additional Related Work

Previous approaches have been proposed for the allocation of circuits to handle specific traffic matrices [13, 65, 79]. Our work is different in several ways. First, in the history-based allocation setting, our formulation takes into consideration the statistical daily traffic variations observed in past measurements and the probability of traffic demands given their statistical distribution of occurrence in past measurements. In our formulation, the allocated circuits do not necessarily provide sufficient circuit capacities for supporting all the traffic matrices captured in the historical data sets. Instead, our problem is formulated as a utility max-min fair circuit allocation problem that aims to maximize the acceptance probability of the expected traffic demand by using the cumulative distribution function over the historical data sets as the objective function. Our solution allocates all available network resources across multiple paths to provide as much headroom as possible. Since our solution does not rely on an online dynamic circuit creation mechanism, there is no need to leave behind network resources for establishing new circuits.

Second, even in the case of our real-time-based allocation setting, our problem formulation allows for the actual traffic in the next period to be different from the current measurement period, and we fully allocate all network resources allow for some fluctuations in traffic rates. This setting is formulated as a weighted max-min fair circuit allocation problem. Our convex optimization approach makes it possible to solve both problems in a unified framework.

There have also been prior work on weighted max-min fair allocation and utility max-min fair allocation for bandwidth allocation problems, but they either only considered the single-path case [19, 68, 64] or provided approximate solutions [4, 22, 23] in the multipath case based on a binary search of achievable utilities. Our approach in this chapter is different in that the problems are solved as a single convex optimization problem, including the modeling of historical traffic distributions as concave functions.

3.7 Conclusion

In this chapter, we considered circuit allocation problems for unified packet and circuit switched networks. We proposed a novel convex optimization framework based on a new destination-based multicommodity flow formulation for the allocation of circuits in such unified networks. In particular, we consider two deployment settings, one based on real-time traffic monitoring, and the other relying upon history-based traffic predictions. In both cases, we formulate global network optimization objectives as concave functions that capture the fair sharing of network capacity among competing traffic flows. The convexity of our problem formulations ensures globally optimal solutions.

3.8 Acknowledgment

Chapter 3, in full, is a reprint of the materials as it appears in Optimization and Engineering, 2019. Yin, Ping; Diamond, Steven; Lin, Bill; Boyd, Stephen. The dissertation author was the primary investigator and author of these papers.

Chapter 4

Strengthening the Positive Effect of Viral Marketing

4.1 Introduction

The study of social network is booming in recent years thanks to the popularity of large online social networking websites. People are interested in how the person-to-person connections within a social network benefitting marketing strategies, and then proposed *viral marketing*, which is based on the diffusion of innovations via the connections in a social network. A key theoretical question in viral marketing is the “seeding” problem, which studies how to optimally select a set of initial adopters. The application scenario which the seeding problem comes from is as follows: Imagine that a firm would like to market a product to a group of users on the social network, but the network is too big such that it is impossible for the firm to target all the users. Therefore, the firm chooses the most influential ones as the initial adopters, and starts an innovation cascade from them using the person-to-person connections. Approaches of this problem is to model it as an optimization problem that aims to maximize the number of users that can be reached by the cascade.

Although extensive works have been conducted in traditional influence maximization problem, most of them agree that the benefit of doing viral marketing is purely increasing in the number of users reached. In other words, it is commonly assumed that the users touched by the cascade are always positive towards the products. However, numerous research works in both marketing and online information dynamics have shown evidences that more users being exposed to a product does not necessarily grant higher benefit. The most influential example of such finding is *Groupon Effect* discovered by [18]. In this work Byers et al. note that Groupon has some negative impact on average Yelp ratings, and they provide some rationales for the phenomenon. One of the key reasons they argue is that when using deal marketplace like Groupon, it attracts a huge amount of customers from many portions of the population, including some groups that are less inclined to like the product. In another example, Aizen et al. [3] investigate the online videos and other media, and find that when some popular blog links to them, the rating drops discontinuously, since more users are driven to the video item including many ones who may not be interested in it. In [48], Kovcs and Sharkey et al. find a similar effect. They survey Goodreads - a book sharing website, and check the books that win prestigious awards. They find that when a book attracts more readers following the announcement, its average rating declines.

Evidences showing that exposing a product to different groups of users may harm the benefit are also found by the studies in Marketing [14] [38] [44]. Berger and Heath in [14] propose that for a consumer, if a product is seen as symbolic of his or her identity, then the consumer often makes different choices from those of others, and even dislikes the product if it is preferred by the majority. In [38], Hu and Van argue that the word-of-mouth effect for a product is independent to the users. They also propose that when promoting products, the viral marketing strategy that boosts the adoption among middle-class customers might diminish the product popularity among higher-income customers.

Take consideration of above effects collectively, different users in different groups have different even opposite views to a product, and there exist some users in the network who incline to dislike the product. Increasing the number of users being exposed to the product is not always leading to higher benefit, since if the marketing cascade reaches the users who tend to dislike the product, they will respond to it negatively, and these negative effects may offset the positive gains brought by viral marketing and suffer the marketing firm. The negative effects can come in different forms, like latent decline on brand loyalty or explicit negative reviews on rating websites.

People may be concerned of how to identify the negative users in the network. This can be achieved by many means. For example, from Amazon review history, by just taking the average review points, we can easily identify the users who are more picky and tend to give harsh reviews. Another more adoptive way is to randomly survey the users and ask their opinions towards the product, and then use cluster algorithms like K-Means to analyze everyone's likelihood of giving bad feedback to the product. We may also take communities into consideration, for instance, sport cars are less likely to be preferred among the community of parents since they prefer more spacious and more solid vehicles to carry their children; It is difficult for the gamers to give positive feedbacks to the light-weight laptops since the gamers often emphasize the impressive gaming performance which is lacked on light-weight laptops. In a word, the identification of negative users is feasible, but it is not the focus of this chapter.

In our chapter, we study on how to integrate the potential positive and negative user response to the marketing cascade into influence propagation model. Based on this model, we consider how to maximize the actual benefit, which is the difference of the total positive response and the total negative response. To be more specific, we make following contribution.

- We modify the classic widely-used influence model - Independent Cascade (IC) model to reflect positive and negative responses. In our model, the users will have initial positive/negative status to indicate their instinctive inclines to like the product. Once being exposed to the cascade, the negative users bring negative impact in terms of negative reviews etc., and they will not spread the cascade any longer. On the contrary, the positive users will like the product and continue propagating the cascade.

- The problem we study is how to choose the initial adopters (seeds) in the network to maximize the actually benefit, i.e., the difference of the total positive response and the total negative response. We call our problem *Strengthening the Positive Effect (SPE)* since it is apparently our goal.

- We check the objective function of *SPE* and find it is non-monotone and non-submodular. What is more, we prove that *SPE* cannot be solved with any positive guarantee unless P=NP. Since generally solving *SPE* with some guarantee is almost impossible, we break our problem into some special cases. We first investigate the unweighted and undirected network, and find an almost optimal algorithm for a special case, and a $(1 - \mathbb{c}_1^p)$-approximation for the general case. For general network, we give a reasonable constraint and devise a $\left[1 - e^{-(1-\mathbb{c}^p)}\right] \frac{1-\epsilon}{1+\epsilon}$-approximation algorithm. Both $\mathbb{c}_1^p$ and $\mathbb{c}^p$ are supermodular curvatures that can be computed polynomially.

- We have tested our solution on real-world social networks of different scales. We verify that comparing with the heuristic, our solution not only provides traceable guarantee, but also outperforms it in terms of running time.

4.2 Related Work

There are many works focusing on influence maximization problem. In [47], Kempe et al. present two widely-used influence cascade models - Linear Threshold (LT) and

Independence Cascade (IC) models. They prove that the influence function is monotone and submodular under both models, which means greedy algorithm has good performance. Since simple greedy proposed in [47] is time consuming, many followup work [50] [21] [35] focus on enhancing the scalability of greedy solutions. In [15], Borgs et al. make a breakthrough. They creatively discover the relation between a set's influence and the probability that a node uniformly chosen at random reaching a node in this set in the graph's transpose. A randomized quasi-linear time algorithm whose guarantee is $(1-1/e-\epsilon)$ has been proposed, and this work opens the door to discovering scalable algorithms of influence related problems on large scale networks. In [77] [76], Tang et al. further optimized the parameters of [15]. Tang et al. in [45] propose a metaheuristic discrete bat algorithm (DBA) based on the collective intelligence of bat population, they also devise a probabilistic greedy-based local search strategy based on network topology to enhance algorithm performance.

Some researchers notice that users in different social groups may have different interests on a product, therefore propose the influence models that include the user evaluation, and devise flexible pricing strategies for all individuals instead of just giving free samples to a representative set. Models of this type are different from what we pursue here. For example, Michael et al. in [46] present a canonical structure to price with network effects, which is a set of consumers with different levels of willingness to pay for a product. In [36] Hartline et al. study revenue maximization, and identify the influence-and-exploit strategy: which is to initially give free item a chosen set of seeds, and then extract revenue from the remaining users by giving them discounts. Arthur et al. [8] consider offering cash-back to encourage a user to spread cascade and giving via coupons or discounts to encourage a user to adopt the product. Lu et al. [53] extend the LT model to incorporate the influence-and-exploit strategy, and show that the expected profit function under their model is submodular but no longer monotone. In [92], Zhu et al. generalize the model proposed in [53]. The authors take both profit and influence into consideration and design effective algorithms. In [63], Pham et al. study the Budgeted Competitive Influence Maximization problem with both budget and time constraints. In their model all competitors are conducting a similar strategy.

There are some works that directly consider negative and positive effects for a marketing cascade propagating within the network [20] [89] [82] [91] [29] [2]. In [20], Chen et al. propose a model where all users have a same chance p to turn negative when receiving the marketing cascade. They prove that the total number of users turning positive is monotone and submodular w.r.t. the number of seeds. Zhang et al. in [89] introduce a parameter called opinion indicator to each user, this indicator keeps updating when the cascade keeps propagating. In the last if the indicator exceeds a threshold then the user becomes positive, otherwise it becomes negative. In [82], Wen et al. propose an analytical model to present the temporal dynamics of spread such as the time people check newly arrived messages or forward them. They found that people's preference and the injection time of the opposing information are critical to the propagation. Khomami et al. in [29] propose a concept called the minimum positive influence dominating set (MPIDS). The authors then prove that by properly choosing the parameters of the algorithm, the probability of finding the MPIDS can be made as close to unity as possible. In [2], Abebe et al. give a model where each node has an initial criticality parameter θ_i which decides if the node is positive or negative to the product. They consider how to choose seeds to maximize the payoff which is the difference of the number of positive nodes minus the number of negative nodes, and give a polynomial solution for the case where there is no limit on the number of seeds.

4.3 Preliminary

4.3.1 Independent Cascade (IC) model

In IC model, the social network is presented as a weighted graph (V, E, ω_E), where the edge weights in ω_E are in interval $(0, 1]$. In the beginning there is a set of nodes which are called the *seeds*. At time 0, only the seeds are *active*. At time t, for each node u that newly becomes *active* at time $t-1$, for each of u's out-edges (u, v), u will try to activate v with the success probability w_{uv} where w_{uv} is the weight of (u, v). Once a node is activated it always keeps active. The cascade terminates until there are no new nodes being activated.

4.3.2 Monotonicity, submodularity, and supermodularity of a set function

Suppose f is a set function on a base set V, then f maps from 2^V to $\mathbb{R}$, where 2^V is the power set of V and $\mathbb{R}$ is the set of real numbers. There are three basic properties to describe set functions: If for any two sets $A \subseteq B$, $f(A) \leq f(B)$, then f is **monotone increasing**; If for any sets $A \subseteq B \subseteq V$ and element $v \in V \setminus B$, $f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B)$, then f is **submodular**; If $-f$ is *submodular*, then f is **supermodular**.

4.3.3 Supermodular curvatures

Suppose that h is a supermodular set function on base V, we define the **supermodular curvature** as

$$\mathbb{C}^p = 1 - \min_{v \in V} \frac{h(v)}{h(V) - h(V \setminus v)}.$$

4.3.4 Maximizing Monotone Submodular+Supermodular Functions

Theorem 4.3.1. *If $f(A)$ is a set function on base V, and f can be expressed as $f_b(A)+f_p(A)$ where f_b is submodular and f_p is supermodular, then for following problem:*

$$\max_{A\subseteq V,|A|\leq k} f(A) \coloneqq f_b(A) + f_p(A), \tag{4.1}$$

the greedy algorithm yields multiplicative guarantee of $[1 - e^{-(1-\mathbb{c}^p)}]$. If $f_b = 0$, i.e., f is purely monotone supermodular, then greedy yields $(1 - \mathbb{c}^p)$, where $\mathbb{c}^p$ is the supermodular curvature of f_p.

4.4 Problem Statement

4.4.1 Diffusion Model

We denote the original social network as a graph (V, E, ω_E), where V is the set of nodes, E is the set of edges, and ω_E is the set of edge weights that are in interval $(0, 1]$. Assume there is a set of negative nodes $\mathsf{N} \in V$ who incline to dislike the product. We call N the **Negative Set**, and $u \in \mathsf{N}$ is called the **negative node**. The classic *IC* model is used as the influence diffusion model, with the exception that the nodes in N will not diffuse the cascade to their neighbors since they dislike the product. Given a seed set S, for each node $v \in V$, we use $p(S, v)$ to denote the probability that v turns *active* to the influence originated from S, and we replace $p(v)$ for $p(S, v)$ when S is clearly stated in the context.

Definition 4.4.1 (Negative Effect)**.** *The negative effect of a seed set S is defined as $\pi^-(S) = \sum_{v\in \mathsf{N}} p(v)$.*

Definition 4.4.2 (Positive Effect)**.** *The positive effect of a seed set S is defined as $\pi^+(S) = \sum_{v \in V \setminus \mathsf{N}} p(v)$.*

Definition 4.4.3 (Strengthening the Positive Effect Problem (***SPE***))**.** *Given a social graph G, a negative set N, and the cost k, to find the seed set $|S| = k$, such that the benefit of S:*

$$\pi(S) = \pi^+(S) - \pi^-(S) \tag{4.2}$$

is maximized.

The *SPE* problem is clearly **NP**-hard. By setting $\mathsf{N} = \emptyset$, *SPE* becomes the classic IC **influence maximization (IM)** problem, which is **NP**-hard proved in [47]. In fact, *SPE* is harder than IM problems in IC model or any other regular diffusion models. Its objective function is non-monotone and non-submodular in the simplest undirected and unweighted graphs, which means traditional algorithm analysis techniques cannot be used to judge the algorithm's performance even on simple graphs.

Proposition 4.4.1. *The objective function $\pi(S)$ of SPE is non-monotone, non-submodular, and non-supermodular.*

We will prove Proposition 4.4.1 in the Appendix. What is more, it can be shown that *SPE* is much harder than many IM problems that can be resolved within factor of $1 - 1/e$, since no non-trivial approximation algorithm is possible to be obtained.

Theorem 4.4.1. *SPE cannot be approximately solved to any positive factor in polynomial time.*

We use Alg. 1 - GreedMax as one the basic techniques when solving *SPE*. GreedMax applies to any set function f.

Algorithm 1: GreedMax

```
   input: a set function $f$ and a cost $k$;
   output: a set $\hat{S}$.
1: $\hat{S} \leftarrow \emptyset, R \leftarrow V$;
2: for $1 \leq i \leq k$ do
3:     $v \leftarrow \arg\max_{v \in R} f(\{v\} \cup \hat{S}) - f(\hat{S})$;
4:     $\hat{S} \leftarrow \{v\} \cup \hat{S}$;
5:     $R \leftarrow R \setminus v$;
6: end for
7: return $\hat{S}$.
```

4.5 Unweighted Undirected Graph

In this section we will try to solve the *SPE* problem on a relatively 'simple' class of social graphs - unweighted and undirected graphs. Unlike IM problem under IC model which is trivial on unweighted and undirected graphs, there is no solution with positive guarantee for *SPE* problem even on this kind of graphs (please refer to the proof of Theorem 4.4.1 in Appendix).

Although it seems very unlikely for us to accurately solve *SPE* in acceptable amount of time, we have two observations to help us devise efficient algorithms:

Observation 1: For any node $v \notin \mathsf{N}$, if there exists a path from the seed set S to v and all nodes on this path are from $V \setminus \mathsf{N}$, then v will be positively activated.

Observation 2: For any node $u \in \mathsf{N}$, if all its neighbors are from N, then deleting u from the graph has no effect on the original *SPE* problem.

Above observations inspire us for Alg. 2 - Prune, which is used to simplify the original graph G. First, if we delete all nodes in N and their associated edges in G and a node u still reaches v in the new graph, then v will definitely be activated by u in G, and vice versa. Further, after deleting N, the new graph may not be connected anymore, and for all the nodes within each connected component, they are all mutually reachable. In other words, the nodes in each connected component are symmetric, choosing any single node as a seed is enough to activate the whole connected component positively. Meanwhile, any node cannot positively activate the nodes in other components, since there is no path consisting entirely of the nodes in $V \setminus \mathsf{N}$. Second, we only add back the nodes in N that connect to some node in $V \setminus \mathsf{N}$ on the original graph G, since if a node in N has no neighbors in $V \setminus \mathsf{N}$, then it will never be activated and has nothing to do with *SPE*'s objective function. We define N' as the set of all the nodes adding back by steps 6 to 9 of Alg. 2. Formally, $\mathsf{N}' = \{v \in \mathsf{N} | \exists u \in V \setminus \mathsf{N}, (u, v) \in E\}$.

Algorithm 2: Prune

```
    input: $G$, a unweighted undirected graph, and $\mathsf{N}$, a negative set;
    output: $G'$, a node weighted undirected graph.
 1: Delete all nodes in $\mathsf{N}$ and their associated edges;
 2: Find the connected components $C_1, C_2, \cdots, C_t$ of the remaining graph;
 3: $U' \leftarrow \emptyset, N' \leftarrow \emptyset$;
 4: for $1 \leq i \leq t$ do
 5:     Contract $C_i$ into a single node $u'_i$ with weight $w_{u'_i} = |C_i|$,
 6:     for $v \in \mathsf{N}$ do
 7:         if $v$ was connected to any node $u \in C_i$ on $G$
 8:             Add an edge between $u'_i$ and $v$, $\mathsf{N}' \leftarrow \mathsf{N}' \cup \{v\}$;
 9:         end if
10:     end for
11: end for
12: $U' \leftarrow \cup_{i=1}^{t} \{u'_i\}$, $V' \leftarrow U' \cup \mathsf{N}'$;
13: $\omega_{V'} \leftarrow$ the weights of nodes in $V'$, $w_v = 1$ for all $v \in \mathsf{N}'$;
14: $E' \leftarrow$ the set of all remaining edges;
15: return $G' = \{V', \omega_{V'}, E'\}$.
```

Since all nodes within a single connected component are symmetric, we further contract each component into a single node with weight, and this weight is the size of the component. We further assign the weight 1 to the remaining nodes in N', then we get the *Pruned Graph.*

Definition 4.5.1 (Pruned Graph)**.** *Given a graph $G = (V, E)$ and a negative set N, the resulting node weighted (bipartite) graph $G' = (V', \omega_{V'}, E')$ after applying Prune algorithm on G is called the **Pruned Graph** of G.*

Note: The pruned graph G' is bipartite.

To see this, first we check the nodes u_i' in U' that coming from contracting C_i, obviously there is no edge between u_i's. Second we check the nodes $v \in \mathsf{N}'$, there is no edge between them since we only add back the edges between v and the nodes in U'. Therefore, G' is bipartite.

On the pruned graph, for $u \in U'$, define $\mathsf{N}'(u) = \{v \in \mathsf{N}' | v \text{ is connected to } u\}$. The *SPE* problem is equivalent to: To find a k-size $S \subseteq U'$, such that:

$$\pi(S) = \sum_{u \in S} w_u - |\cup_{u \in S} \mathsf{N}'(u)| \tag{4.3}$$

is maximized.

Proposition 4.5.1. *The objective function $\pi(S)$ of SPE on pruned graph G' is supermodular.*

Although *SPE* problem cannot be approximately solved to any positive ratio on unweighted and undirected graphs, it is still possible and inspiring to get some performance bounds when the graphs satisfy some certain constrains. We use S^* to denote the optimal solution. The following theorem tells us that the benefit obtained by GreedMax is only $(k-1)$ smaller than the optimal if the pruned graph is a forest. This is a small margin, since the cost k is often very tiny comparing to the size of the network and the benefit π.

Theorem 4.5.1. *If the pruned graph G' is a forest, then GreedMax is guaranteed to obtain a solution $\hat{S}$ such that*

$$\pi(\hat{S}) \geq \pi(S^*) - (k-1).$$

The bound is tight.

Comparing with theorem 4.5.1, theorem 4.5.2 applies to more graphs since its constraint is more relaxed. If the negative set N is not too large nor concentrated too much, then theorem 4.5.2 is very likely to apply.

Theorem 4.5.2. *If $\forall u \in U', w_u \geq |\mathsf{N}'(u)|$, then GreedMax is guaranteed to obtain a solution $\hat{S}$ such that*

$$\pi(\hat{S}) \geq (1 - \mathbb{c}_1^p)\pi(S^*),$$

where $\mathbb{c}_1^p$ is the supermodular curvature of $\pi(S)$ in (4.3).

4.6 General Weighted Directed Graphs

After discussing how to solve *SPE* on unweighted and undirected graphs, in this section we investigate *SPE* on general (edge) weighted and directed graphs.

In fact, *SPE* on general graphs is tough. An intuitive solution would be to use Monte Carlo simulation to estimate $\pi(S)$, and call GreedMax using this estimation as the input f. This method applies to any general graph. However, it works slowly and its performance is impossible to analyze since Monte Carlo simulation is hard to trace.

We would like to design some scalable algorithm for *SPE*, and if possible, the algorithm with bound is greatly preferred. However, according to Theorem 4.4.1, there is no algorithm with positive guarantee. To make things easier, we consider giving the *SPE* a constraint, which is $|\mathsf{N}| \leq k - 1$. The reason why we make this constraint is three-fold: 1) The negative set N is considered to be the stubborn negative users in the network, and when we start doing viral marketing, not many users are aware of the product. Therefore it is reasonable to assume that not many nodes are firmly against it. 2) The cost k is decided by the viral marketing executor, and the executor can adjust it according to his or her estimation of $|\mathsf{N}|$. 3) Under this constraint, it is obvious that for any seed set S that $|S| = k$, the benefit $\pi(S) \geq 1$, and this is a nice property that we can use to design scalable algorithms.

4.6.1 The restricted *SPE* where $|\mathsf{N}| \le k-1$

We will design a randomized algorithm for the restricted *SPE* discussed above, and our idea comes from the concept called *reverse reachable set* which is initiated in [15]. In *SPE*, in order to investigate the existence of negative nodes more accurately, we modify reverse reachable set and introduce the idea called *positive and negative reverse reachable sets*.

Definition 4.6.1 (Positive and Negative Reverse Reachable Sets)**.** *A Reverse Reachable (RR) set for a node $v \in V$ is generated by first sampling a graph h from the triggering model, and then taking the set of nodes in h that can reach v. If $v \notin \mathsf{N}$, then this is a positive RR set, denoted by RR$^+$, otherwise it is a negative RR set, denoted by RR$^-$.*

Alg. 3 - *RR Set Sampling (RRS)* is used to randomly generate a RR$^+$ or RR$^-$ set. This sampling algorithm is important because we want the sets to be truly 'random', and step 1 guarantees it. Note that steps 4 and 6 of Alg. 3 is the triggering process of IC model.

Algorithm 3: RRS (RR Set Sampling)

```
    input: $G$, a general edge weighted graph, and $\mathsf{N}$, a negative set;
    output: $H$, a set of nodes.
 1: Randomly choose a node $u$ with probability $\frac{1}{n}$, $H \leftarrow H \cup \{u\}$;
 2: do
 3:     Pick a unchecked node $u \in H$;
 4:     for $e = (v, u)$ that connected to $u$ do
 5:         if $v$ is unchecked then
 6:             Add $v$ to $H$ randomly with probability $w_e$;
 7:         end if
 8:     Mark $u$ as checked;
 9:     end for
10: while $H$ changes;
11: if $u \in \mathsf{N}$ then return $H$ as a RR$^-$ set;
12: else return $H$ as a RR$^+$ set;
13: end if
```

RR$^-$ sets help us to identify the nodes which are highly possible to activate the negative nodes, and we should avoid these nodes that frequently appear in RR$^-$ sets. Meanwhile, the nodes frequently appearing in RR$^+$ sets have a good potential to bring positive effect. To quantify the idea, denote $\mathcal{H} = \mathcal{H}^+ \cup \mathcal{H}^-$ the superset of all RR sets generated, where $\mathcal{H}^+$ and $\mathcal{H}^-$ are the supersets of all RR$^+$ sets and RR$^-$ sets respectively. Suppose $\mathcal{H} = \{H_1, H_2, \cdots, H_\rho\}$, then we define a random variable $x_i \in \{+1, 0, -1\}, 1 \leq i \leq \rho$. For a given node set S, if $S \cap H_i = \emptyset$ then $x_i = 0$, if $S \cap H_i \neq \emptyset$ and $H_i \in \mathcal{H}^+$ then $x_i = +1$, else if $S \cap H_i \neq \emptyset$ and $H_i \in \mathcal{H}^-$ then $x_i = -1$.

Based on the definition of x_i, for the set S and the superset $\mathcal{H}$ of ρ generated RR sets, we denote $F_{\mathcal{H}^+}(S)$ as the probability that S intersects $\mathcal{H}^+$, $F_{\mathcal{H}^-}(S)$ as the probability that S intersects $\mathcal{H}^-$, and $F_{\mathcal{H}}(S)$ as the difference of $F_{\mathcal{H}^+}(S)$ subtracting with $F_{\mathcal{H}^-}(S)$, i.e.,

$$\begin{aligned} F_{\mathcal{H}}(S) &= F_{\mathcal{H}^+}(S) - F_{\mathcal{H}^-}(S) \\ &= \frac{|\{H \in \mathcal{H}^+ | H \cap S \neq \emptyset\}|}{|\mathcal{H}^+|} - \frac{|\{H \in \mathcal{H}^- | H \cap S \neq \emptyset\}|}{|\mathcal{H}^-|}, \end{aligned}$$

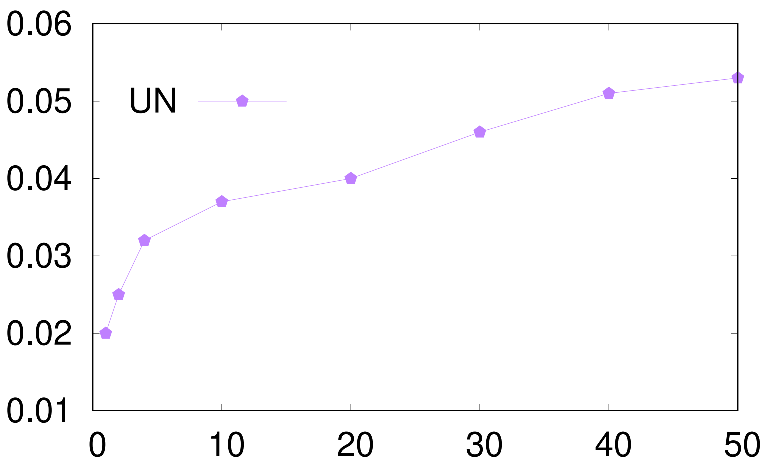
then

$$F_{\mathcal{H}}(S) = \frac{1}{\rho} \cdot \sum_{i=1}^{\rho} x_i. \tag{4.5}$$

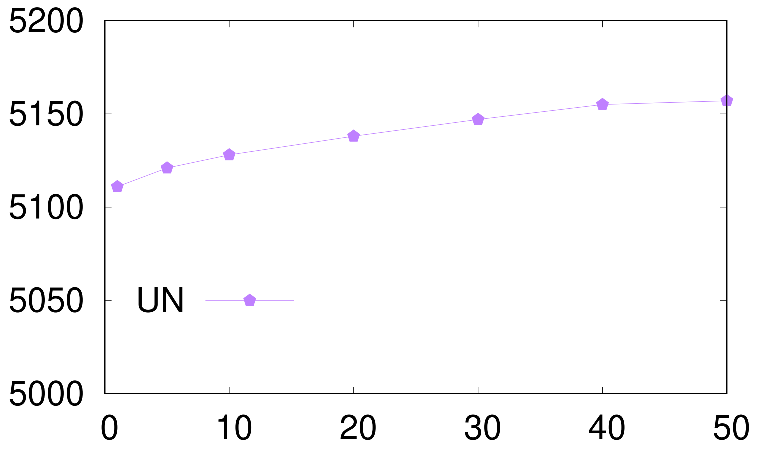
Lemma 4.6.1. *For a set $S \subseteq V$, and any superset $\mathcal{H}$ of RR sets generating by Alg. 3,*

$$\mathbb{E}\left[F_{\mathcal{H}}(S)\right] = \frac{\pi(S)}{n}.$$

Note that $F_{\mathcal{H}}(S)$ is the statistic we calculate from a randomly generated superset $\mathcal{H}$, and it is easy to think that the more sample RR sets we have in $\mathcal{H}$, the closer our calculated $F_{\mathcal{H}}(S)$ is to its expectation. However, generating more sample RR sets leads

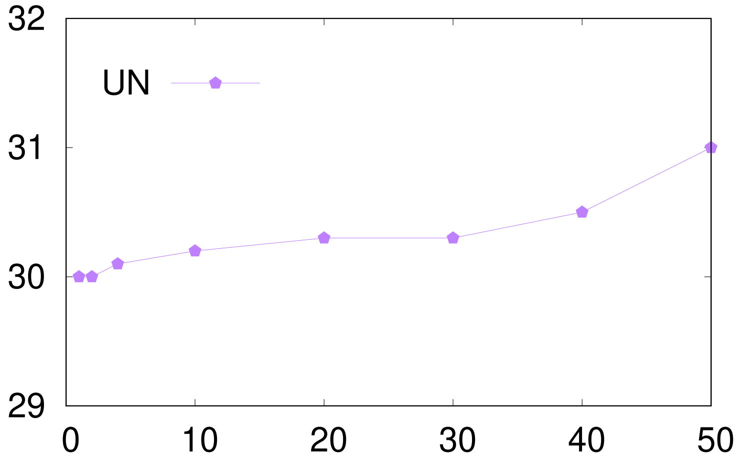


(a) Running Time (min)

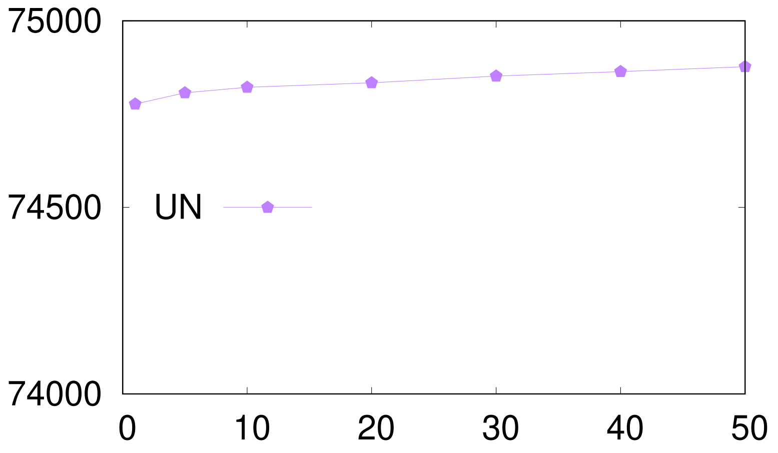


(b) Benefit

Figure 4.1: Unweighted Undirected GRQC.

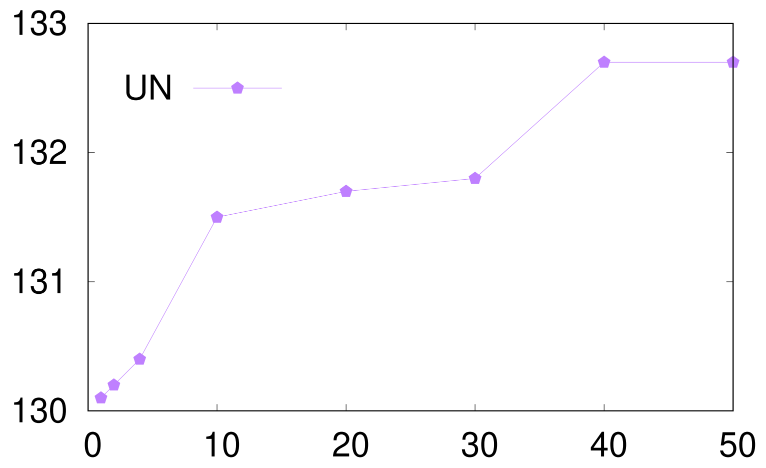


(a) Running Time (min)

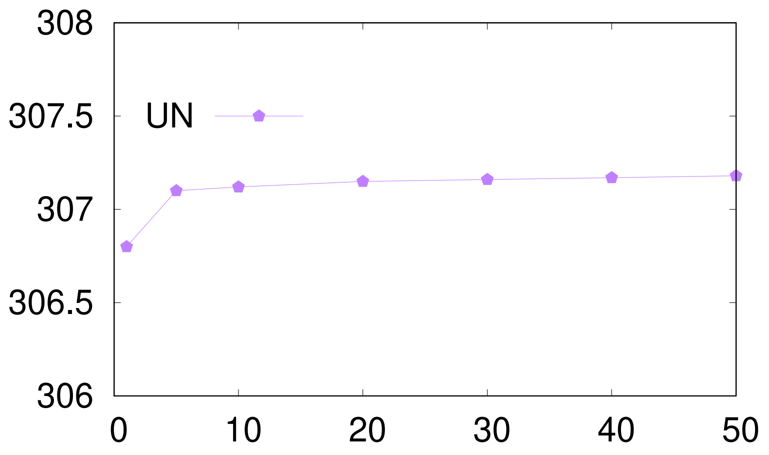


(b) Benefit

Figure 4.2: Unweighted Undirected Epinions.



(a) Running Time (min)



(b) Benefit (K)

Figure 4.3: Unweighted Undirected Orkut.

to the higher algorithmic time complexity, and following lemma tells us that how many sample RR sets are required for a reasonable guarantee.

Lemma 4.6.2. *For $\epsilon, \delta \in (0,1)$, let*

$$\rho' = \frac{2n \cdot \ln(1/\delta)}{\epsilon^2}, \tag{4.6}$$

then on a superset $\mathcal{H}$ containing more than ρ' RR sets, for any k-size set S, each of following inequalities stands with at least $(1-\delta)$ probability:

$$n \cdot F_{\mathcal{H}}(S) \leq (1+\epsilon) \cdot \pi(S),$$

$$(1-\epsilon) \cdot \pi(S) \leq n \cdot F_{\mathcal{H}}(S).$$

Lemma 4.6.3. *For $\epsilon, \delta \in (0,1)$, let*

$$\rho = \frac{2n \cdot (\ln 2 + \ln(1/\delta))}{\epsilon^2}. \tag{4.7}$$

Suppose the superset $\mathcal{H}$ contains more than ρ RR sets. If the function $F_{\mathcal{H}}(S)$ is monotone increasing, then Alg. 1 returns a $(1-e^{-(1-\mathbb{c}^p)})\frac{1-\epsilon}{1+\epsilon}$ solution with $1-\delta$ probability, where $\mathbb{c}^p$ is the supermodular curvature of $-F_{\mathcal{H}^-}(S)$.

Algorithm 4: BM (Benefit Maximization)

```
   input: $G, \mathsf{N},$ and $k,$ the number of seeds;
   output: a set containing $k$ seeds.
1: while $|\mathcal{H}| \leq \rho$ do
   // $\rho$ is defined in (4.7)
2:     $\mathcal{H} \leftarrow \mathcal{H} \cup \mathrm{RRS}(G)$;
3: end while
4: return GreedMax($F_{\mathcal{H}}$, $k$).
```

Note that ρ in (4.7) is a constant that only depends on n and the error parameters

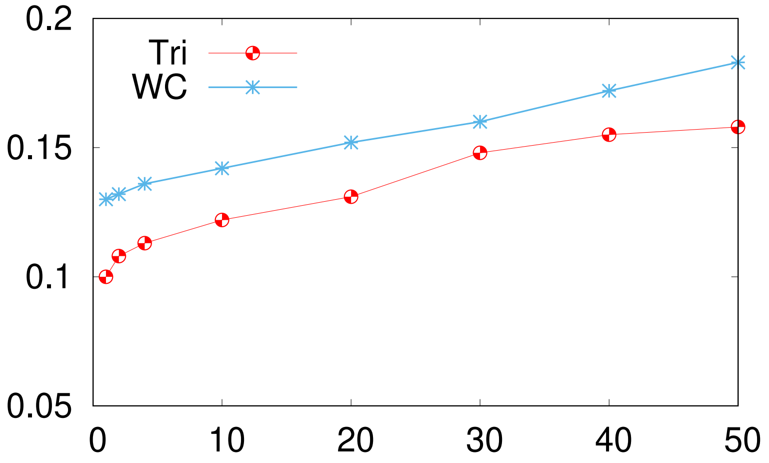
δ and ϵ. Denote S^* the optimal seed set, and OPT the benefit of S^*, in many randomized algorithms for viral marketing, if ρ is the number of RR sets needed then ρ is often related to the OPT. However, in *SPE* we can no longer associate ρ with OPT, since the performance ratio of GreedMax is now $1 - e^{-(1-\mathbb{c}^p)}$, and this is not a constant. For the resulting set S, we have no fixed guarantee of how close $\pi(S)$ of the solution S is to OPT, and the number of RR sets that needed to approximate $\pi(S)$ is actually related to $\pi(S)$. Therefore OPT is not sufficient to guarantee the performance bound. We can only use the fact which is $\pi(S) \geq 1$ for any k-size set S to compute the minimum number of RR sets. Following summative theorem gives the guarantee for our designed algorithms.

Theorem 4.6.1. *For given error $\epsilon, \delta \in (0,1)$, Alg. 4 (BM) gives Problem* SPE *a solution $\hat{S}$ such that $\pi(\hat{S}) \geq \left[1 - e^{-(1-\mathbb{c}^p)}\right] \frac{1-\epsilon}{1+\epsilon} \cdot \pi(S^*)$ with probability at least $1-\delta$ if $|\mathsf{N}| \leq k-1$ and $F_{\mathcal{H}}$ is monotone increasing, where $\mathbb{c}^p$ is the supermodular curvature of $-F_{\mathcal{H}^-}$. Alg. 4 runs in $O(nm \cdot \ln \frac{1}{\delta} \cdot \epsilon^{-2})$ expected time.*

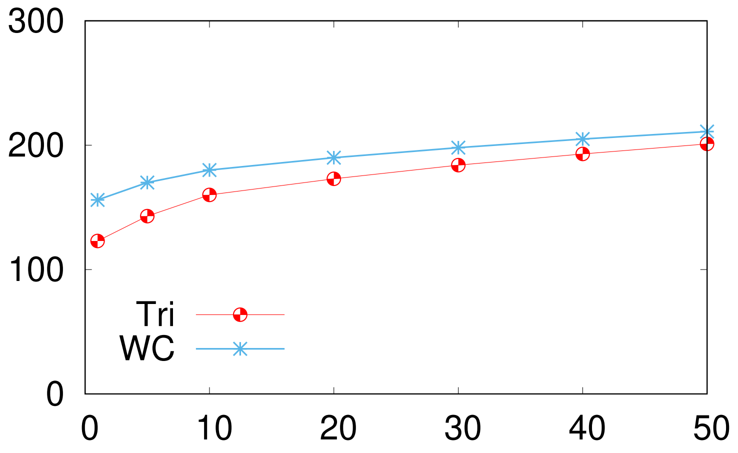
4.7 Experiments

Datasets. We select three datasets whose sizes are from thousands to hundreds of millions edges: (Arxiv) GRQC (General Relativity and Quantum Cosmology) [51] is a citation network, Epinions [66] and Orkut [84] are online social networks. Epinions.com was a general consumer review site. At Epinions, visitors can read new and old reviews about a variety of items which can help them to decide on a purchase. Orkut, on the other hand, was a social networking website. The service was designed to help users meet new and old friends and maintain existing relationships. Table 4.1 presents the statistics of our employed datasets. It can be seen that Epinions has over 130K nodes and 840K edges, while Orkut has over 3M nodes and 110M edges.

Generating propagation probabilities. We need to assign each edge a weight

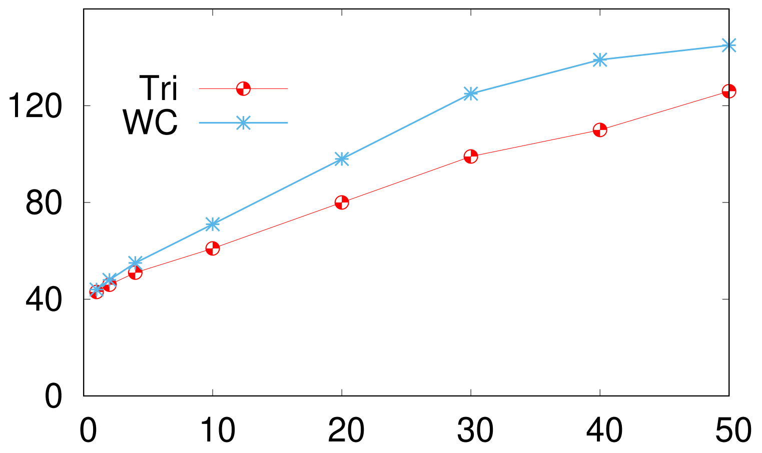


(a) Running Time (min)

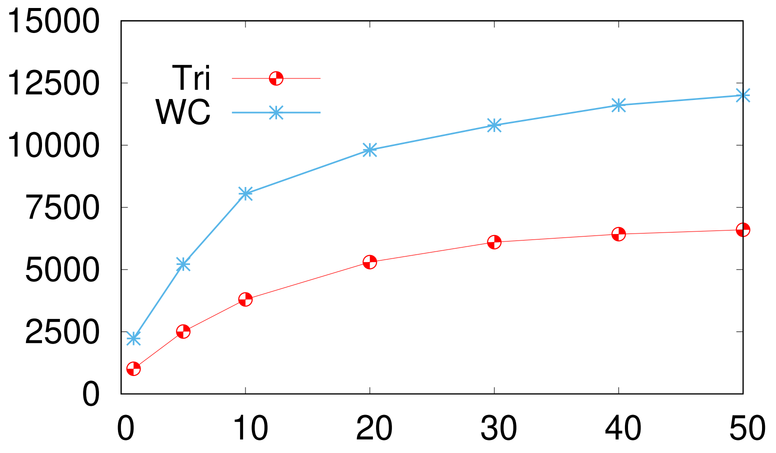


(b) Benefit

Figure 4.4: General GRQC.

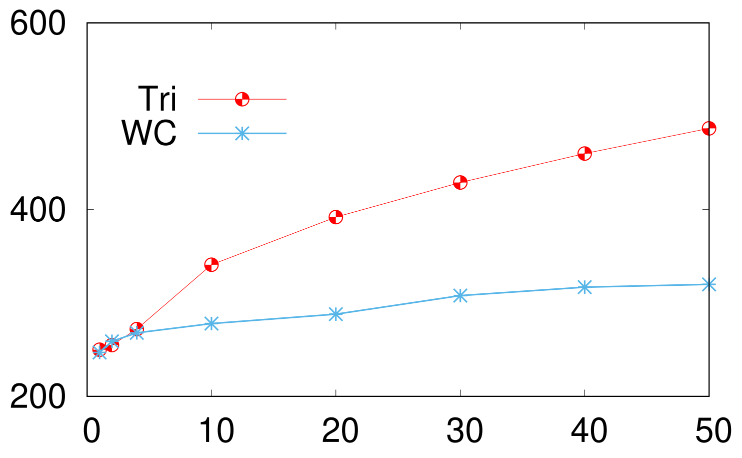


(a) Running Time (min)

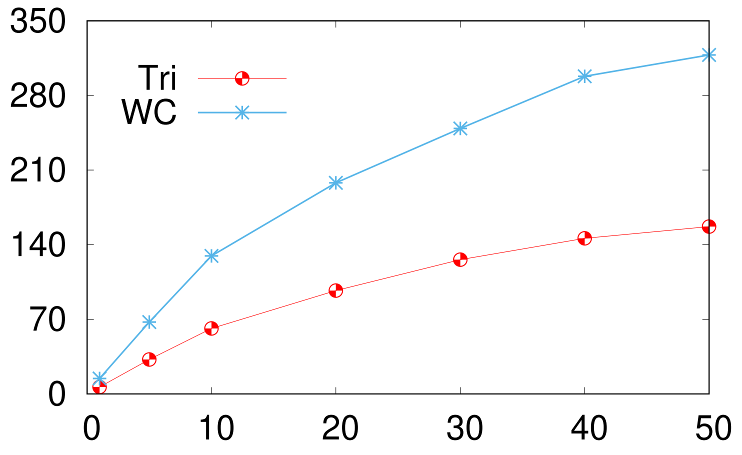


(b) Benefit

Figure 4.5: General Epinions.



(a) Running Time (min)



(b) Benefit (K)

Figure 4.6: General Orkut.

Table 4.1: Dataset Statistics

	GRQC	Epinions	Orkut
#Nodes	5K	132K	3M
#Edges	29K	841K	117M
Avg. deg.	5.5	13.4	37.8

as the influence propagation probability. In the case of general graph, we use two methods to generate the propagation probabilities: 1) **Weighted Cascade (WC)** model, in which the probability w_{uv} of edge (u, v) equals $1/d(v)$, where $d(v)$ is the in-degree of v; 2) **Trivalency (Tri)** model, in which the probability for each edge is uniformly at random chosen from the set $\{0.1, 0.01, 0.001\}$. Both above methods are classic ways to determine the propagation probabilities adopted in many studies like [47], [21] etc.

Other Parameters. There are other parameters that play important roles in our simulation. One of them is N, the number of negative users. In the case of unweighted and undirected graph, we alway set $|\mathsf{N}| = 100$. In the case of general graph, we set $|\mathsf{N}| = k - 1$, where k is the number of seeds to be identified. When testing Alg. 4, we set the error bounds $\epsilon = 0.8$ and $\delta = \frac{1}{\ln n}$.

Table 4.2: Running Time Comparison

	GRQC	Epinions	Orkut
BM, $k = 10$, Tri	0.12m	61m	341m
MC, $k = 10$, Tri	43m	511m	N/A
BM, $k = 10$, WC	0.14m	71m	278m
MC, $k = 10$, WC	51m	723m	N/A

Result Analysis. Before we explain the experimental results, note that for each figure in Figures 4.1-4.3 and 4.4-4.6, the horizontal axis is k, the number of seeds, and the vertical axis is the running time or benefit (indicated by its caption).

Figures 4.1-4.3 plot *BM*'s performance on unweighted and undirected graphs. It can be seen that the benefit a single seed mines is not too far away from that of tens of seeds. To be more specific, the benefit a single seed mines on GRQC achieves 99% of that 50 seeds mine on the same network. On Epinions, a single seed mines almost 99.8% benefit that 50 seeds mine, while on Orkut, a single seed achieves 99.8% benefit that 50 seeds mine and 2 seeds achieves 99.99% benefit that 50 seeds mine. The running time, however, increases faster when we increase the number of seeds. To determine a single seed it costs 0.02 minutes while it costs more than 0.05 minutes on GRQC. On Epinions, it costs 30 minutes to identify a seed while it costs 31 minutes to identify 50 seeds. It costs 130 minutes to determine a seed and 132.6 minutes to determine 50 seeds on Orkut. We also test our algorithm extensively on general graphs. We use *BM* to identify up to 50 seeds on general graphs, and the results are plotted in Fig. 4.4-4.6. *BM* only determines the top seeds, but it does not need or give the actual benefit. In order to test the benefit as well, we use Alg. 5 shown in the end of the Appendix. Our algorithm *BM* determines the top seeds on graphs containing hundreds of millions edges within several hours. It can be seen that on general graphs, under WC model, comparing with unit graphs, increasing the number of seeds raises the benefit more evidently. The benefit a single seed mines on GRQC achieves 56% of that 50 seeds mine on the same network. On Epinions, a single seed mines almost 15% benefit that 50 seeds mine, while on Orkut, a single seed achieves 4% benefit that 50 seeds mine. Note that there is currently no good heuristic that works for *SPE* besides *Monte Carlo (MC)*. Therefore, we compare *BM* with *MC* on the running time to identify $k = 10$ seeds and plot the result in table 4.2. Note that it takes too long for *MC* to find the seeds on Orkut dataset and we put N/A as the running time. We didn't compare the benefit since the *MC* took too much time when we set k greater than 10. At last we check the curvature $\mathbb{c}^p$ on different datasets, and the result is plotted in Table 4.3. It can be seen that the larger the dataset is, the smaller the curvature is, which means the algorithm's theoretical performance guarantee is better.

Table 4.3: Curvature

	GRQC	Epinions	Orkut
Unweighted undirected graph	0.98	0.82	0.37
General graph	0.99	0.98	0.98

4.8 Conclusion

Many studies confirm that there always exist some users in the network who tend to be negative to the product, therefore the traditional viral marketing strategy of exposing a product to as many users in the network as possible may not always bring the best outcome. To consider this issue, we study the optimization problem called *Strengthening the Positive Effect (SPE)*.

The objective function of *SPE* is non-monotone and non-submodular, and cannot be solved approximately within any positive guarantee. We further investigate *SPE* in depth. When the social network is unweighted and undirected, we discover that the almost optimal polynomial solution exists on a special case, and also find another approximate result for a more general case. When the network is general with edge weights and edge directions, we set a reasonable constraint and also design an approximate algorithm. Our algorithms are tested on real-world datasets and their effectiveness is testified.

4.9 Acknowledgment

Chapter 4, in full, is a reprint of the material as it appears in IEEE 39th International Conference on Distributed Computing Systems, 2019. Zhu, Yuqing; Yin, Ping; Li, Deying; Lin, Bill. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Conclusion

In this dissertation, we have demonstrated several new approaches for network performance improvements to combat relentless growth in Internet traffic. In Chapter 2, we have proposed a series of new backpressure-based routing algorithms by means of incremental expansion of routing choices as needed for packet switching network. In Chapter 3, we have proposed a novel convex optimization framework based on a new destination-based multicommodity flow formulation for the allocation of circuits in unified packet and circuit switched networks. Finally, in Chapter 4, we have presented a new approach to the influence spreading optimization problem for large social networks that considers the negative impact of negative users.

Chapter 6

Appendix

A.1 Proof for Chapter 2

In this section, we use fluid models to prove that the L-BP, A-BP, O-BP and E-BP algorithms are all throughput optimal.

A.1.1 Modeling and Assumptions

At each timeslot, each node needs to make a schedule to transmit data in the network. Let $\mathcal{S}$ be the set of all possible schedules. Each schedule $\beta = (\beta_{ab}^{(c)} : a, b, c = 1, \ldots, N) \in \mathcal{S}$ is a vector in $\mathbb{Z}^{N\times N\times N}$, where $\beta_{ab}^{(c)}$ gives the amount of commodity c data sent from node a to node b via link (a, b) under schedule β. It is assumed that $\beta_{cn}^{(c)} = 0$ for all $\beta \in \mathcal{S}$, $c = 1, \ldots, N$ and $n = 1, \ldots, N$. It is also assumed that $\mathcal{S}$ is *monotone* in the following sense: if $\beta \in \mathbb{Z}^{N\times N\times N}$ and there exists $\beta' \in \mathcal{S}$ such that $\beta \leq \beta'$, then $\beta \in \mathcal{S}$. This is because if β' is a valid schedule and we decrease the amount of data sent in some of the links, the resulted schedule must also be a valid one.

We define $\mathcal{S}(t) \subset \mathcal{S}$ as the set of valid schedules given the scheduling strategy and the systems status at timeslot t. We assume that $\mathcal{S}(t)$ maintains the monotonicity of $\mathcal{S}$, i.e., $\beta \in \mathcal{S}(t)$ if there exists $\beta' \in \mathcal{S}(t)$ such that $\beta \leq \beta'$. A necessary constraint on $\mathcal{S}(t)$ is that we must have

$$\sum_{b=1}^{N} \beta_{nb}^{(c)} \leq Q_n^{(c)}(t) \tag{A.1}$$

for all $\beta \in \mathcal{S}(t)$ and $n, c = 1, \ldots, N$. For each schedule $\beta \in \mathcal{S}$, we define a "collapsed" schedule $\gamma(\beta) = (\gamma_n^{(c)}(\beta) : n, c = 1, \ldots, N) \in \mathbb{Z}^{N \times N}$ where $\gamma_n^{(c)}(\beta) = \sum_{b=1}^{N} \beta_{nb}^{(c)} - \sum_{a=1}^{N} \beta_{an}^{(c)}$ is the speed that schedule β empties the backlog in queue $Q_n^{(c)}$. Let $\Gamma_{\mathcal{S}} \subset \mathbb{Z}^{N \times N}$ be the set of all possible collapsed schedules given $\mathcal{S}$. Let $< \Gamma_{\mathcal{S}} >$ be the convex hull of $\Gamma_{\mathcal{S}}$. Assume each link (a, b), $a, b = 1, \ldots, N$, has a finite maximal transmission speed, i.e., $\exists\ R \geq 0$ such that $\beta_{ab}^{(c)} \leq R$ for all $\beta \in \mathcal{S}$ and $a, b, c = 1, \ldots, N$. Then both $\mathcal{S}$ and $\Gamma_{\mathcal{S}}$ are finite sets.

To analyze the stability of our schemes, we first define a family of generalized max-weighted scheduling schemes as follows. Define the weight $W(\beta, Q(t))$ of schedule β given queue length $Q(t)$ as

$$W(\beta, Q(t)) \triangleq \sum_{a=1}^{N} \sum_{b=1}^{N} \sum_{c=1}^{N} \beta_{ab}^{(c)} [Q_a^{(c)}(t) - Q_b^{(c)}(t)]$$

At each timeslot, a schedule $\beta \in \mathcal{S}$ that solves the following optimization problem will be selected for activation.

$$\begin{aligned} \max_{\beta} \quad & W(\beta, Q(t)) \\ s.t. \quad & \beta \in \mathcal{S}(t) \end{aligned}$$

The baseline (original) backpressure scheme, our modified route-expanding and semi-oblivious BP schemes all belong to this family of generalized max-weighted scheduling schemes. The only difference among these schemes is the definition of the valid schedule set $\mathcal{S}(t)$ at each timeslot.

Let $\mathcal{S}^{BP}(t)$, $\mathcal{S}^{L}(t)$, $\mathcal{S}^{A}(t)$, $\mathcal{S}^{O}(t)$, $\mathcal{S}^{E}(t)$ be the set of valid schedules at timeslot t for the baseline backpressure policy, the L-BP policy, the A-BP policy, the O-BP policy and the E-BP policy respectively.

We have

$$\mathcal{S}^{BP}(t) = \left\{ \beta \in \mathcal{S} \,\middle|\, \sum_{b=1}^{N} \beta_{nb}^{(c)} \leq Q_n^{(c)}(t), \forall n, c = 1, \ldots, N \right\}$$

$$\mathcal{S}^{L}(t) \subseteq \mathcal{S}^{BP}(t)$$

$$\mathcal{S}^{A}(t) \subseteq \mathcal{S}^{BP}(t)$$

$$\mathcal{S}^{O}(t) \subseteq \mathcal{S}^{BP}(t)$$

$$\mathcal{S}^{E}(t) \subseteq \mathcal{S}^{BP}(t)$$

$$\mathcal{S}^{L}(t) \supseteq \left\{ \beta \in \mathcal{S}^{BP}(t) \,\middle|\, \beta_{nb}^{(c)} = 0 \text{ if } Q_n^{(c)}(t) < L_{max}, n, b, c = 1, \ldots, N \right\} \tag{A.2}$$

$$\mathcal{S}^{A}(t) \supseteq \left\{ \beta \in \mathcal{S}^{BP}(t) \,\middle|\, \beta_{nb}^{(c)} = 0 \text{ if } E_n^{(c)}(t) < A_{max}, n, b, c = 1, \ldots, N \right\} \tag{A.3}$$

$$\mathcal{S}^{O}(t) \supseteq \left\{ \beta \in \mathcal{S}^{BP}(t) \,\middle|\, \beta_{nb}^{(c)} = 0 \text{ if } Z_n^{(j)}(t) < H_{max}, n, b, c = 1, \ldots, N, j = 1, \ldots, M_n \right\} \tag{A.4}$$

$$\mathcal{S}^{E}(t) \supseteq \left\{ \beta \in \mathcal{S}^{BP}(t) \,\middle|\, \beta_{nb}^{(c)} = 0 \text{ if } Z_n^{(j)}(t) < H_{max}, n, b, c = 1, \ldots, N, j = 1, \ldots, M_n \right\} \tag{A.5}$$

In (A.3), $E_n^{(c)}(t)$ represents the age of the head packet at time t, which is the number of timeslots that the head packet has waited in $Q_n^{(c)}$. In (A.4) and (A.5), M_n is the number of nodes neighbor to node n.

Let $T_\beta(t)$, $\beta \in \mathcal{S}$, be the cumulative number of time slots that schedule β was employed by timeslot t. From (A.1) , we have

$$D_{ab}^{(c)}(t) = \sum_{\beta \in \mathcal{S}} \sum_{\ell=1}^{t} \beta_{ab}^{(c)} \cdot (T_\beta(\ell) - T_\beta(\ell - 1)) \tag{A.6}$$

To formulate the fluid model, we extend the above discrete time functions to the continuous time domain. Specifically, for $t \in [0, +\infty)$, we define

$$\begin{aligned}
&A_n^{(c)}(t) = A_n^{(c)}(\lfloor t \rfloor) \qquad n, c = 1, \ldots, N \\
&Q_n^{(c)}(t) = Q_n^{(c)}(\lfloor t \rfloor) \qquad n, c = 1, \ldots, N \\
&D_{ab}^{(c)}(t) = D_{ab}^{(c)}(\lfloor t \rfloor) + (t - \lfloor t \rfloor)(D_{ab}^{(c)}(\lceil t \rceil) - D_{ab}^{(c)}(\lfloor t \rfloor)), \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad a, b, c = 1, \ldots, N \\
&T_\beta(t) = T_\beta(\lfloor t \rfloor) + (t - \lfloor t \rfloor)(T_\beta(\lceil t \rceil) - T_\beta(\lfloor t \rfloor)) \qquad \beta \in \mathcal{S}
\end{aligned}$$

where $\lfloor t \rfloor$ is the largest integer that is smaller than or equal to t and $\lceil t \rceil$ is the smallest integer that is larger than or equal to t.

Assume the arrival process $A(t)$ satisfies a strong law of large numbers (SLLN), i.e., there exists a constant arrival rate matrix $\lambda = (\lambda_n^{(c)} : n, c = 1, \ldots, N)$, such that, with probability one,

$$\lim_{t \to \infty} \frac{A_n^{(c)}(t)}{t} = \lambda_n^{(c)} \qquad \forall\ n, c = 1, \ldots, N \tag{A.7}$$

Without loss of generality, we assume that $\lambda_c^{(c)} = 0$ for all $c = 1, \ldots, N$ for simplicity.

A.1.2 Basic Fluid Model Equations

We now investigate the stochastic process $(Q(t), D(t), T(t))$, where

$$\begin{aligned} Q(t) &\triangleq \left\{Q_n^{(c)}(t) \,\middle|\, n, c = 1, 2, \ldots, N\right\} \\ D(t) &\triangleq \left\{D_{ab}^{(c)}(t) \,\middle|\, a, b, c = 1, 2, \ldots, N\right\} \\ T(t) &\triangleq \left\{T_\beta(t) \,\middle|\, \beta \in \mathcal{S}\right\} \end{aligned}$$

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be the probability space that this stochastic process is defined on, where Ω is the sample space, $\mathcal{F}$ is a σ-field on Ω, and $\mathbb{P}$ is the probability measure on $(\Omega, \mathcal{F})$. We shall sometimes use the notations $Q(\cdot, \omega)$, $D(\cdot, \omega)$ and $T(\cdot, \omega)$ to explicitly denote the dependency on the sample path $\omega \in \Omega$.

Now, for each $r > 0$, we define fluid scaled processes

$$\left(\hat{Q}^r(t, \omega), \hat{D}^r(t, \omega), \hat{T}^r(t, \omega)\right) \triangleq \frac{1}{r}\left(Q(rt, \omega), D(rt, \omega), T(rt, \omega)\right)$$

Proposition A.1.2.1 (Fluid Model)**.** *For each sample path $\omega \in \Omega$ satisfying (A.7) and any sequence $\{r_n\}$ with $r_n \to \infty$, there exists a subsequence $\{r_{n_k}\}$ and continuous functions $(\hat{Q}, \hat{D}, \hat{T})$ with $\hat{Q}(0) = 0$, such that*

$$\left(\hat{Q}^{r_{n_k}}(\cdot, \omega), \hat{D}^{r_{n_k}}(\cdot, \omega), \hat{T}^{r_{n_k}}(\cdot, \omega)\right) \longrightarrow \left(\hat{Q}, \hat{D}, \hat{T}\right) \quad u.o.c \quad as \;\; k \to \infty \tag{A.8}$$

where the convergence is uniform on compact sets (u.o.c). The three-tuple $(\hat{Q}, \hat{D}, \hat{T})$ is said to be a fluid limit path of the system. It satisfies the following fluid model equations

$$\hat{Q}_n^{(c)}(t) = \lambda_n^{(c)} t + \sum_{a=1}^{N} \hat{D}_{an}^{(c)}(t) - \sum_{b=1}^{N} \hat{D}_{nb}^{(c)}(t)$$

$$n, c = 1, \ldots, N \text{ and } n \neq c \tag{A.9}$$

$$\hat{Q}_c^{(c)}(t) = 0 \qquad c = 1, \ldots, N \tag{A.10}$$

$$\hat{D}_{ab}^{(c)}(t) = \sum_{\beta \in \mathcal{S}} \beta_{ab}^{(c)} \hat{T}_\beta(t) \qquad a, b, c = 1, \ldots, N \tag{A.11}$$

$$\sum_{\beta \in \mathcal{S}} \hat{T}_\beta(t) = t \tag{A.12}$$

$$\hat{Q}_n^{(c)}(t) \geq 0 \qquad n, c = 1, \ldots, N$$

$$\hat{T}_\beta(0) = 0, \quad \hat{T}_\beta(\cdot) \textit{ is non-decreasing} \qquad \beta \in \mathcal{S}$$

$$\hat{T}_\beta(t) - \hat{T}_\beta(s) \leq t - s \textit{ for } 0 \leq s < t \qquad \beta \in \mathcal{S}$$

The proof of Proposition 1 is somewhat standard. We refer the reader to [26].

A.1.3 Main Results

We first give the formal definition of throughput optimilities.

Definition A.1.3.1 (Rate Stability). *We say the system is rate stable, if with probability one,*

$$\lim_{t \to \infty} \frac{\sum_{n=1}^{N} D_{nc}^{(c)}(t)}{t} = \sum_{n=1}^{N} \lambda_n^{(c)} \qquad c = 1, \ldots, N \tag{A.13}$$

for any arrival process satisfying (A.7).

Note that the left-hand-side of (A.13) is actually the long-run average rate of commodity c packets that *depart* from the network, while the right-hand-side is the long-run average rate of commodity c packets that *arrive* to the network. In other words, the system is guaranteed to achieve 100% throughput whenever it is rate stable.

Definition A.1.3.2 (Throughput Optimal). *We say an arrival process $A(t)$ is admissible if its arrival rate matrix λ belongs to $< \Gamma_{\mathcal{S}} >$. The system is said to be throughput optimal if it is rate stable under any admissible arrival process.*

The main results of this section are stated as follows.

Theorem A.1.3.1. *The system is throughput optimal when working under the L-BP policy.*

Theorem A.1.3.2. *The system is throughput optimal when working under the A-BP policy.*

Theorem A.1.3.3. *The system is throughput optimal when working under the O-BP or E-BP policy.*

We will give the proof of these results using fluid model in the rest of this section. More specifically, we'll first prove that their fluid model is weakly stable as defined in Definition A.1.3.3 and the stability of the original system is then guaranteed by Proposition A.1.3.1. The detailed proof of Proposition A.1.3.1 is provided in Appendix A.1.3.

Definition A.1.3.3 (Weak Fluid Stability)**.** *The fluid model is said to be weakly stable if for each fluid limit path with $\hat{Q}(0) = 0$ we have $\hat{Q}(t) = 0$ for all $t \geq 0$.*

Proposition A.1.3.1. *For a given arrival rate matrix λ, the system is rate stable if the corresponding fluid model is weakly stable.*

Proof of Proposition A.1.3.1

Proof. From our assumptions in Section A.1.1, we know that $\beta_{cn}^{(c)} = 0$ for all $\beta \in \mathcal{S}$ and $n, c = 1, \ldots, N$. Then by (A.11), we know that $\hat{D}_{cn}^{(c)}(t) = 0$ for all $n, c = 1, \ldots, N$. This equality will be used in justifying Equation (A.14) below.

Denote $\mathcal{N} = \{1, \ldots, N\}$ as the set of nodes in the network. From (A.9) and (A.10), for each commodity c, we have

$$\begin{aligned}\sum_{n=1}^{N} \hat{Q}_n^{(c)}(t) &= \sum_{n \in \mathcal{N} \setminus \{c\}} \hat{Q}_n^{(c)}(t) \\ &= \sum_{n \in \mathcal{N} \setminus \{c\}} \left(\lambda_n^{(c)} t + \sum_{a=1}^{N} \hat{D}_{an}^{(c)}(t) - \sum_{b=1}^{N} \hat{D}_{nb}^{(c)}(t) \right)\end{aligned}$$

$$
\begin{aligned}
&= \sum_{n\in\mathcal{N}\setminus\{c\}} \lambda_n^{(c)} t + \sum_{n\in\mathcal{N}\setminus\{c\}} \sum_{a=1}^{N} \hat{D}_{an}^{(c)}(t) \\
&\quad - \sum_{n\in\mathcal{N}\setminus\{c\}} \sum_{b=1}^{N} \hat{D}_{nb}^{(c)}(t) \\
&= \sum_{n=1}^{N} \lambda_n^{(c)} t + \sum_{n\in\mathcal{N}\setminus\{c\}} \sum_{a=1}^{N} \hat{D}_{an}^{(c)}(t) - \sum_{n=1}^{N}\sum_{b=1}^{N} \hat{D}_{nb}^{(c)}(t) \qquad \text{(A.14)} \\
&= \sum_{n=1}^{N} \lambda_n^{(c)} t - \sum_{c=1}^{N} \hat{D}_{nc}^{(c)}(t) \qquad \text{(A.15)}
\end{aligned}
$$

Equation (A.14) stands because (1) $\lambda_c^{(c)} = 0$ for $c = 1, \ldots, N$ from our assumptions in in Section A.1.1; and (2) $\hat{D}_{cn}^{(c)}(t) = 0$ for $n, c = 1, \ldots, N$.

If the fluid model is weakly stable, by Definition A.1.3.3, we know that $\sum_{n=1}^{N} \hat{Q}_n^{(c)}(t) = 0$ for all $t \geq 0$. Then by (A.15), we have

$$
\sum_{c=1}^{N} \hat{D}_{nc}^{(c)}(t) = \sum_{n=1}^{N} \lambda_n^{(c)} t
$$

Thus

$$
\sum_{c=1}^{N} \hat{D}_{nc}^{(c),r}(t,\omega) \to \sum_{n=1}^{N} \lambda_n^{(c)} t \quad u.o.c \quad as \ r \to \infty
$$

In particular, $\sum_{c=1}^{N} \hat{D}_{nc}^{(c),r}(1,\omega) \to \sum_{n=1}^{N} \lambda_n^{(c)}$ as $r \to \infty$ or

$$
\lim_{r\to\infty} \frac{\sum_{n=1}^{N} D_{nc}^{(c)}(r)}{r} = \sum_{n=1}^{N} \lambda_n^{(c)}
$$

which is actually the same to (A.13), proving the proposition. □

A.1.4 Proof of Theorems A.1.3.1-A.1.3.3

We first present a lemma that will be used in the proof of Theorems A.1.3.1 - A.1.3.3.

Lemma A.1.4.1. *The fluid model defined in Proposition A.1.2.1 is weakly stable if each* *of the fluid limit paths satisfies the following fluid equation.*

$$\text{For each } \beta \in \mathcal{S},\ \frac{d}{dt}\hat{T}_\beta(t) = 0 \ \text{ if } W(\beta, \hat{Q}(t)) < \max_{\alpha \in \mathcal{S}} W(\alpha, \hat{Q}(t)) \tag{A.16}$$

The intuition behind Lemma A.1.4.1 is that a scheduling policy is throughput optimal if, when the system backlog is very large, the weight of the schedule it generates is roughly the same with the weight of the schedule generated by the baseline backpressure policy. The proof of Lemma A.1.4.1 is similar to the proof of Theorem 1 in [26]. For completeness, we produce a full proof in Appendix A.1.4.

It is then sufficient to prove that each of the fluid limit paths satisfies Eq. (A.16) if the system works under the L-BP, A-BP, O-BP or E-BP policy. Intuitively, this is true because when the backlog in the system is very large, the total backlog (and the total weight of the schedule) in the system would be dominated by the Phase II queues (as the queue lengths of the Phase I queues are always bounded by a constant). In other words, the contribution of Phase I queues is negligible when calculating the weight of a schedule. On the other hand, as Phase II queues always execute the baseline backpressure policy in our schemes, the weight of the generated schedule should be close to the weight of the schedule generated by the baseline backpressure policy. The formal proofs of Theorems A.1.3.1-A.1.3.3 are similar to the proof of Lemma 4 in [27]. For completeness, we produce a full proof in Appendices A.1.4-A.1.4.

Proof of Lemma A.1.4.1

Proof. The proof is similar to the proof of Theorem 1 in [26]. For completeness, we produce a full proof here.

It is sufficient to show that the fluid model is weakly stable for any arrival rate matrix λ that belongs to $< \Gamma_{\mathcal{S}} >$. Fix an arrival rate matrix λ that belongs to $< \Gamma_{\mathcal{S}} >$.

Suppose $(\hat{Q}, \hat{D}, \hat{T})$ is a fluid limit path with $|\hat{Q}(0)| = 0$. Define a Lyapunov function

$$f(t) = \sum_{n=1}^{N} \sum_{c=1}^{N} \left(\hat{Q}_n^{(c)}(t) \right)^2$$

We have $f(0) = 0$, $f(t) \geq 0$ and $f(t) = 0 \Leftrightarrow \hat{Q}(t) = 0$ for all $t > 0$. It is then sufficient to prove that such that $f(t) = 0$ for all $t \geq 0$.

Since λ belongs to $< \Gamma_{\mathcal{S}} >$, there exist constants $p_\beta \in [0, 1]$, $\beta \in \mathcal{S}$, such that

$$\lambda_n^{(c)} = \sum_{\beta \in \mathcal{S}} p_\beta \cdot \gamma_n^{(c)}(\beta) = \sum_{\beta \in \mathcal{S}} p_\beta \left(\sum_{b=1}^{N} \beta_{nb}^{(c)} - \sum_{a=1}^{N} \beta_{an}^{(c)} \right), \qquad c = 1, \ldots, N$$

$$\sum_{\beta \in \mathcal{S}} p_\beta \leq 1 \tag{A.17}$$

Let $W_{max}(\hat{Q}(t)) = \max_{\beta \in \mathcal{S}} W(\beta, \hat{Q}(t))$. For $t \geq 0$ we have

$$\begin{aligned}
W_{max}(\hat{Q}(t)) &\geq \sum_{\beta \in \mathcal{S}} p_\beta W(\beta, \hat{Q}(t)) \\
&= \sum_{\beta \in \mathcal{S}} p_\beta \sum_{a=1}^{N} \sum_{b=1}^{N} \sum_{c=1}^{N} \beta_{ab}^{(c)} [\hat{Q}_a^{(c)}(t) - \hat{Q}_b^{(c)}(t)] \\
&= \sum_{\beta \in \mathcal{S}} p_\beta \sum_{n=1}^{N} \sum_{c=1}^{N} \hat{Q}_n^{(c)}(t) \left(\sum_{b=1}^{N} \beta_{nb}^{(c)} - \sum_{a=1}^{N} \beta_{an}^{(c)} \right) \\
&= \sum_{n=1}^{N} \sum_{c=1}^{N} \hat{Q}_n^{(c)}(t) \cdot \sum_{\beta \in \mathcal{S}} p_\beta \left(\sum_{b=1}^{N} \beta_{nb}^{(c)} - \sum_{a=1}^{N} \beta_{an}^{(c)} \right) \\
&= \sum_{n=1}^{N} \sum_{c=1}^{N} \hat{Q}_n^{(c)}(t) \cdot \lambda_n^{(c)}
\end{aligned} \tag{A.18}$$

Let t be a fixed value such that $\hat{Q}(\cdot)$ is differentiable at t. Let $\mathcal{S}'$ be the set of schedules β such that $W(\beta, \hat{Q}(t)) = W_{max}(\hat{Q}(t))$. Then we have $\frac{d}{dt} W(\beta, \hat{Q}(t)) = \frac{d}{dt} W_{max}(\hat{Q}(t))$ for

$\beta \in \mathcal{S}'$ (see proof of Lemma 3.2 of [28]). By (A.12) and (A.16), we have

$$\sum_{\beta \in \mathcal{S}'} \frac{d}{dt}\hat{T}_\beta(t) = 1$$

It follows that

$$\begin{aligned}
&\sum_{n=1}^{N}\sum_{c=1}^{N}\hat{Q}_n^{(c)}(t)\left(\sum_{b=1}^{N}\frac{d}{dt}\hat{D}_{nb}^{(c)}(t) - \sum_{a=1}^{N}\frac{d}{dt}\hat{D}_{an}^{(c)}(t)\right)\\
&= \sum_{n=1}^{N}\sum_{c=1}^{N}\hat{Q}_n^{(c)}(t)\Bigg(\sum_{b=1}^{N}\sum_{\beta\in\mathcal{S}'}\beta_{nb}^{(c)}\frac{d}{dt}\hat{T}_\beta(t)\\
&\qquad\qquad - \sum_{a=1}^{N}\sum_{\beta\in\mathcal{S}'}\beta_{an}^{(c)}\frac{d}{dt}\hat{T}_\beta(t)\Bigg)\\
&= \sum_{\beta\in\mathcal{S}'}\frac{d}{dt}\hat{T}_\beta(t)\sum_{n=1}^{N}\sum_{c=1}^{N}\hat{Q}_n^{(c)}(t)\left(\sum_{b=1}^{N}\beta_{nb}^{(c)} - \sum_{a=1}^{N}\beta_{an}^{(c)}\right)\\
&= \sum_{\beta\in\mathcal{S}'}\frac{d}{dt}\hat{T}_\beta(t)\sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{c=1}^{N}\beta_{ab}^{(c)}[\hat{Q}_a^{(c)}(t) - \hat{Q}_b^{(c)}(t)]\\
&= \sum_{\beta\in\mathcal{S}'}\frac{d}{dt}\hat{T}_\beta(t)\cdot W(\beta,\hat{Q}(t))\\
&= W_{max}(\beta,\hat{Q}(t))\sum_{\beta\in\mathcal{S}'}\frac{d}{dt}\hat{T}_\beta(t)\\
&= W_{max}(\beta,\hat{Q}(t))
\end{aligned}$$

Thus,

$$\begin{aligned}
\frac{d}{dt}f(t) &= 2\sum_{n=1}^{N}\sum_{c=1}^{N}\hat{Q}_n^{(c)}(t)\cdot\frac{d}{dt}\hat{Q}_n^{(c)}(t)\\
&= 2\sum_{n=1}^{N}\sum_{c=1}^{N}\hat{Q}_n^{(c)}(t)\left(\lambda_n^{(c)} + \sum_{a=1}^{N}\frac{d}{dt}\hat{D}_{an}^{(c)}(t) - \sum_{b=1}^{N}\frac{d}{dt}\hat{D}_{nb}^{(c)}(t)\right)\\
&= 2\sum_{n=1}^{N}\sum_{c=1}^{N}\hat{Q}_n^{(c)}(t)\lambda_n^{(c)}
\end{aligned}$$

$$
\begin{aligned}
&- 2\sum_{n=1}^{N}\sum_{c=1}^{N}\hat{Q}_n^{(c)}(t)\left(\sum_{b=1}^{N}\frac{d}{dt}\hat{D}_{nb}^{(c)}(t) - \sum_{a=1}^{N}\frac{d}{dt}\hat{D}_{an}^{(c)}(t)\right)\\
&= 2\left(\sum_{n=1}^{N}\sum_{c=1}^{N}\hat{Q}_n^{(c)}(t)\lambda_n^{(c)} - W_{max}(\hat{Q}(t))\right)\\
&\leq 0
\end{aligned}
$$

In other words, we have $\frac{d}{dt}f(t) \leq 0$ for almost every t such that f is differentiable at t. Since $f(0) = 0$, it follows that $f(t) = 0$ for all $t \geq 0$ (see, for example, the proof of Lemma 1 of [26]), proving the theorem. $\square$

Proof of Theorem A.1.3.1

Proof. It's sufficient to prove that each of the fluid limit paths satisfies Eq. (A.16) if the system works under the L-BP policy. The proof is similar to the proof of Lemma 4 in [27]. For completeness, we produce a full proof here.

Suppose $(\hat{Q}, \hat{D}, \hat{T})$ is a fluid limit path. Fix a sample path $\omega \in \Omega$ such that (A.7) and (A.8) hold. There exists a sequence $\{r_k\}$ with $r_k \to \infty$ as $k \to \infty$, such that

$$
\begin{aligned}
&\left(\hat{Q}^{r_k}(\cdot,\omega),\hat{D}^{r_k}(\cdot,\omega),\hat{T}^{r_k}(\cdot,\omega)\right)\\
&\qquad\to \left(\hat{Q},\hat{D},\hat{T}\right) \quad u.o.c \quad as\ k \to \infty
\end{aligned}
\tag{A.19}
$$

Fix a time $t \geq 0$. Define $\mathcal{A}(\hat{Q}(t)) = \{(a, c) \,|\, \hat{Q}_a^{(c)}(t) > 0\}$. Let $\tilde{\alpha} = \text{argmax}_{\alpha\in\mathcal{S}} W(\alpha, \hat{Q}(t))$. Define $\tilde{\beta}$ via

$$
\tilde{\beta}_{ab}^{(c)} = \begin{cases} \tilde{\alpha}_{ab}^{(c)} & \text{if } (a,c) \in \mathcal{A}(\hat{Q}(t)) \\ 0 & \text{otherwise} \end{cases}
$$

We have $\tilde{\beta} \in \mathcal{S}$ since $\tilde{\beta} \leq \tilde{\alpha}$. We now prove that $W(\tilde{\beta}, \hat{Q}(t)) = \max_{\alpha\in\mathcal{S}} W(\alpha, \hat{Q}(t))$. Note that

$$\begin{aligned}
W(\tilde{\alpha}, \hat{Q}(t)) &= \sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{c=1}^{N} \tilde{\alpha}_{ab}^{(c)}[\hat{Q}_a^{(c)}(t) - \hat{Q}_b^{(c)}(t)] \\
&= \sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{c=1}^{N} \tilde{\alpha}_{ab}^{(c)}[\hat{Q}_a^{(c)}(t) - \hat{Q}_b^{(c)}(t)] \cdot 1_{\hat{Q}_a^{(c)}(t)=0} \\
&+ \sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{c=1}^{N} \tilde{\alpha}_{ab}^{(c)}[\hat{Q}_a^{(c)}(t) - \hat{Q}_b^{(c)}(t)] \cdot 1_{\hat{Q}_a^{(c)}(t)>0} \\
&= \sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{c=1}^{N} \tilde{\alpha}_{ab}^{(c)}[0 - \hat{Q}_b^{(c)}(t)] \cdot 1_{\hat{Q}_a^{(c)}(t)=0} \\
&+ \sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{c=1}^{N} \tilde{\alpha}_{ab}^{(c)}[\hat{Q}_a^{(c)}(t) - \hat{Q}_b^{(c)}(t)] \cdot 1_{\hat{Q}_a^{(c)}(t)>0} \\
&\leq \sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{c=1}^{N} \tilde{\alpha}_{ab}^{(c)}[\hat{Q}_a^{(c)}(t) - \hat{Q}_b^{(c)}(t)] \cdot 1_{\hat{Q}_a^{(c)}(t)>0} \\
&= \sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{c=1}^{N} \tilde{\beta}_{ab}^{(c)}[\hat{Q}_a^{(c)}(t) - \hat{Q}_b^{(c)}(t)] \cdot 1_{\hat{Q}_a^{(c)}(t)>0} \\
&= \sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{c=1}^{N} \tilde{\beta}_{ab}^{(c)}[\hat{Q}_a^{(c)}(t) - \hat{Q}_b^{(c)}(t)]
\end{aligned}$$

Then we must have

$$W(\tilde{\beta}, \hat{Q}(t)) = W(\tilde{\alpha}, \hat{Q}(t)) = \max_{\alpha \in \mathcal{S}} W(\alpha, \hat{Q}(t)).$$

Fix a schedule $\beta \in \mathcal{S}$ with $W(\beta, \hat{Q}(t)) < W(\tilde{\beta}, \hat{Q}(t))$. There exists a constant $\epsilon > 0$ such that

$$W(\tilde{\beta}, \hat{Q}(t)) - W(\beta, \hat{Q}(t))\rangle \geq \epsilon, \hat{Q}_a^{(c)}(t) > \epsilon \text{ for } (a, c) \in \mathcal{A}(\hat{Q}(t))$$

By the continuity of $\hat{Q}(\cdot)$, there exists $\tau > 0$ such that for each $s \in [t-\tau, t+\tau]$

$$\begin{aligned} W(\tilde{\beta}, \hat{Q}(s)) - W(\beta, \hat{Q}(s)) &\geq \frac{\epsilon}{2} \\ \hat{Q}_a^{(c)}(s) > \frac{\epsilon}{2} \qquad &\text{for } (a,c) \in \mathcal{A}(\hat{Q}(s)) \end{aligned}$$

Let R be the maximal link speed all over the network as defined in Section A.1.1. By (A.19), there exists $K > 0$ such that, for any $k > K$ we have $\frac{\epsilon}{4} r_k > \max(L_{max}, NR)$ and for each $s \in [t-\tau, t+\tau]$

$$\begin{aligned} \Bigg| \Big(W(\tilde{\beta}, \hat{Q}^{r_k}(s)) - W(\beta, \hat{Q}^{r_k}(s)) \Big) \\ - \Big(W(\tilde{\beta}, \hat{Q}(s)) - W(\beta, \hat{Q}(s)) \Big) \Bigg| \leq \frac{\epsilon}{4} \end{aligned}$$

$$\left| \hat{Q}_a^{(c),r_k}(s) - \hat{Q}_a^{(c)}(s) \right| \leq \frac{\epsilon}{4}, \text{ for } (a,c) \in \mathcal{A}(\hat{Q}(t))$$

Thus for $k > K$ and each $s \in [t-\tau, t+\tau]$, we have

$$\begin{aligned} W(\tilde{\beta}, \hat{Q}^{r_k}(s)) - W(\beta, \hat{Q}^{r_k}(s)) &\geq \frac{\epsilon}{4} \\ \hat{Q}_a^{(c),r_k}(s) \geq \frac{\epsilon}{4} \qquad &\text{for } (a,c) \in \mathcal{A}(\hat{Q}(t)) \end{aligned}$$

Therefore, for each time $s \in [(t-\tau)r_k, (t+\tau)r_k]$, we have

$$W(\tilde{\beta}, Q(s,\omega)) > W(\beta, Q(s,\omega)) \tag{A.20}$$

$$Q_a^{(c)}(s,\omega) \geq \frac{\epsilon}{4} r_k > \max(L_{max}, NR) \quad \text{for } (a,c) \in \mathcal{A}(\hat{Q}(t)) \tag{A.21}$$

Condition (A.21) implies that schedule $\tilde{\beta}$ only serves queue that has queue backlog larger than $\max(L_{max}, NR)$ throughout time interval $[(t-\tau)r_k, (t+\tau)r_k]$ and the queues it serves all have sufficient backlog to send. By (A.2), it must be a valid schedule under the L-BP policy throughout time interval $[(t-\tau)r_k, (t+\tau)r_k]$. By (A.20), the weight of schedule β is always less than that of $\tilde{\beta}$, and thus should never be employed throughout time interval $[(t-\tau)r_k, (t+\tau)r_k]$. Therefore, for any $u_1 \leq u_2$, $u_1, u_2 \in [(t-\tau)r_k, (t+\tau)r_k]$ we have

$$T_\beta(u_2, \omega) - T_\beta(u_1, \omega) = 0$$

Thus, for any $u_1, u_2 \in [(t-\tau), (t+\tau)]$ with $u_1 \leq u_2$, we have

$$T_\beta(u_2 r_k, \omega) - T_\beta(u_1 r_k, \omega) = 0$$

i.e.,

$$\hat{T}_\beta^{r_k}(u_2, \omega) - \hat{T}_\beta^{r_k}(u_1, \omega) = 0$$

Taking the limit as $k \to \infty$, we have

$$\hat{T}_\beta(u_2) - \hat{T}_\beta(u_1) = 0$$

for any $u_1, u_2 \in [(t-\tau), (t+\tau)]$ with $u_1 \leq u_2$, from which (A.16) follows, proving the lemma. □

Proof of Theorem A.1.3.2

Proof. It is sufficient to prove that each of the fluid limit paths satisfies Eq. (A.16) if the system works under the A-BP policy. The proof is quite similar to that of Theorem A.1.3.1.

Similarly, suppose $(\hat{Q}, \hat{D}, \hat{T})$ is a fluid limit path. Fix a sample path $\omega \in \Omega$ such that (A.7) and (A.8) hold. There exists a sequence $\{r_k\}$ with $r_k \to \infty$ as $k \to \infty$, such that

$$\begin{aligned}&\Big(\hat{Q}^{r_k}(\cdot, \omega), \hat{D}^{r_k}(\cdot, \omega), \hat{T}^{r_k}(\cdot, \omega)\Big)\\ &\qquad \to \Big(\hat{Q}, \hat{D}, \hat{T}\Big) \quad u.o.c \quad as \;\; k \to \infty \end{aligned} \tag{A.22}$$

Fix a time $t \geq 0$. Define $\mathcal{A}(\hat{Q}(t)) = \{(a, c) \,|\, \hat{Q}_a^{(c)}(t) > 0\}$. Let $\tilde{\alpha} = \operatorname{argmax}_{\alpha \in \mathcal{S}} W(\alpha, \hat{Q}(t))$. Define $\tilde{\beta}$ via

$$\tilde{\beta}_{ab}^{(c)} = \begin{cases} \tilde{\alpha}_{ab}^{(c)} & \text{if } (a, c) \in \mathcal{A}(\hat{Q}(t)) \\ 0 & \text{otherwise} \end{cases}$$

We have $\tilde{\beta} \in \mathcal{S}$ and $W(\tilde{\beta}, \hat{Q}(t)) = \max_{\alpha \in \mathcal{S}} W(\alpha, \hat{Q}(t))$.

Fix a schedule $\beta \in \mathcal{S}$ with $W(\beta, \hat{Q}(t)) < W(\tilde{\beta}, \hat{Q}(t))$. There exists a constant $\epsilon > 0$ such that

$$W(\tilde{\beta}, \hat{Q}(t)) - W(\beta, \hat{Q}(t))\rangle \geq \epsilon$$

and $\hat{Q}_a^{(c)}(t) > \epsilon$ for $(a, c) \in \mathcal{A}(\hat{Q}(t))$. By the continuity of $\hat{Q}(\cdot)$, there exists $\tau > 0$ such that for each $s \in [t - \tau, t + \tau]$

$$\begin{aligned} &W(\tilde{\beta}, \hat{Q}(s)) - W(\beta, \hat{Q}(s)) \geq \frac{\epsilon}{2} \\ &\hat{Q}_a^{(c)}(s) > \frac{\epsilon}{2} \qquad \text{for } (a, c) \in \mathcal{A}(\hat{Q}(s)) \end{aligned}$$

Let R be the maximal link speed all over the network as defined in Section A.1.1. By (A.22), there exists $K > 0$ such that, for any $k > K$ we have $\frac{\epsilon}{4} r_k > \max(RA_{max}, RN)$, $\frac{\tau}{2} r_k > A_{max}$ and for each $s \in [t - \tau, t + \tau]$

$$\Big|\Big(W(\tilde{\beta}, \hat{Q}^{r_k}(s)) - W(\beta, \hat{Q}^{r_k}(s))\Big) - \Big(W(\tilde{\beta}, \hat{Q}(s)) - W(\beta, \hat{Q}(s))\Big)\Big| \leq \frac{\epsilon}{4}$$

$$\Big|\hat{Q}_a^{(c),r_k}(s) - \hat{Q}_a^{(c)}(s)\Big| \leq \frac{\epsilon}{4} \qquad \text{for } (a,c) \in \mathcal{A}(\hat{Q}(t))$$

Thus for $k > K$ and each $s \in [t-\tau, t+\tau]$, we have

$$W(\tilde{\beta}, \hat{Q}^{r_k}(s)) - W(\beta, \hat{Q}^{r_k}(s)) \geq \frac{\epsilon}{4}$$
$$\hat{Q}_a^{(c),r_k}(s) \geq \frac{\epsilon}{4} \qquad \text{for } (a,c) \in \mathcal{A}(\hat{Q}(t))$$

Therefore, for each time $s \in [(t-\tau)r_k, (t+\tau)r_k]$, we have

$$W(\tilde{\beta}, Q(s)) > W(\beta, Q(s)) \tag{A.23}$$

$$Q_a^{(c)}(s) \geq \frac{\epsilon}{4} r_k > \max(RA_{max}, RN), \text{ for } (a,c) \in \mathcal{A}(\hat{Q}(t)) \tag{A.24}$$

Condition (A.24) implies that for $(a,c) \in \mathcal{A}(\hat{Q}(t))$, the length of queue $Q_a^{(c)}$ is always larger than RA_{max}. Thus the delay of the head-of-line packet in $Q_a^{(c)}$ is always larger than A_{max} throughout time interval $[(t-\tau)r_k + A_{max}, (t+\tau)r_k]$. Since $\frac{\tau}{2}r_k > A_{max}$, we have $[(t-\frac{\tau}{2})r_k, (t+\frac{\tau}{2})r_k] \subset [(t-\tau)r_k + A_{max}, (t+\tau)r_k]$. In other words, schedule $\tilde{\beta}$ only serves queue with delay of head-of-line packets larger than A_{max} throughout time interval $[(t-\frac{\tau}{2})r_k, (t+\frac{\tau}{2})r_k]$ and the queues it serves all have sufficient backlog to send. By (A.3), it must be a valid schedule under the A-BP policy throughout time interval $[(t-\frac{\tau}{2})r_k, (t+\frac{\tau}{2})r_k]$. By (A.23), the weight of schedule β is always less than that of $\tilde{\beta}$, and thus should never be employed throughout time interval $[(t-\frac{\tau}{2})r_k,(t+\frac{\tau}{2})r_k]$. Therefore, for any $u_1 \leq u_2$, $u_1, u_2 \in [(t-\frac{\tau}{2})r_k,(t+\frac{\tau}{2})r_k]$, we have

$$T_\beta(u_2,\omega) - T_\beta(u_1,\omega) = 0$$

Therefore, for any $u_1, u_2 \in [(t-\frac{\tau}{2}),(t+\frac{\tau}{2})]$ with $u_1 \leq u_2$, we have

$$T_\beta(u_2 r_k,\omega) - T_\beta(u_1 r_k,\omega) = 0$$

i.e.,

$$\hat{T}_\beta^{r_k}(u_2,\omega) - \hat{T}_\beta^{r_k}(u_1,\omega) = 0$$

Taking the limit as $k \to \infty$, we have

$$\hat{T}_\beta(u_2) - \hat{T}_\beta(u_1) = 0$$

for any $u_1, u_2 \in [(t-\frac{\tau}{2}),(t+\frac{\tau}{2})]$ with $u_1 \leq u_2$, from which (A.16) follows, proving the theorem. □

Proof of Theorem A.1.3.3

Proof. We first prove the throughput optimality of O-BP. It is sufficient to prove that each of the fluid limit paths satisfies the Eq. (A.16) if the system works under the O-BP policy. The proof is quite similar to that of Theorem A.1.3.1.

Similarly, suppose $(\hat{Q},\hat{D},\hat{T})$ is a fluid limit path. Fix a sample path $\omega \in \Omega$ such that (A.7) and (A.8) hold. There exists a sequence $\{r_k\}$ with $r_k \to \infty$ as $k \to \infty$, such that

$$\begin{aligned}\Big(\hat{Q}^{r_k}(\cdot,\omega), &\hat{D}^{r_k}(\cdot,\omega), \hat{T}^{r_k}(\cdot,\omega)\Big) \\ &\to \Big(\hat{Q}, \hat{D}, \hat{T}\Big) \quad u.o.c \quad as\ k \to \infty \end{aligned} \tag{A.25}$$

Fix a time $t \geq 0$. Define $\mathcal{A}(\hat{Q}(t)) = \{(a,c) \,|\, \hat{Q}_a^{(c)}(t) > 0\}$. Let $\tilde{\alpha} = \text{argmax}_{\alpha \in \mathcal{S}} W(\alpha, \hat{Q}(t))$. Define $\tilde{\beta}$ via

$$\tilde{\beta}_{ab}^{(c)} = \begin{cases} \tilde{\alpha}_{ab}^{(c)} & \text{if } (a,c) \in \mathcal{A}(\hat{Q}(t)) \\ 0 & \text{otherwise} \end{cases}$$

We have $\tilde{\beta} \in \mathcal{S}$ and $W(\tilde{\beta}, \hat{Q}(t)) = \max_{\alpha \in \mathcal{S}} W(\alpha, \hat{Q}(t))$.

Fix a schedule $\beta \in \mathcal{S}$ with $W(\beta, \hat{Q}(t)) < W(\tilde{\beta}, \hat{Q}(t))$. Similar to the proof of Theorem A.1.3.1, we can prove that there exist constants $\tau > 0$ and $K > 0$ such that, for any $k > K$ and time $s \in [(t-\tau)r_k, (t+\tau)r_k]$, we have

$$W(\tilde{\beta}, Q(s,\omega)) > W(\beta, Q(s,\omega)) \tag{A.26}$$

$$Q_a^{(c)}(s,\omega) > \max(H_{max}, NR) \quad \text{for } (a,c) \in \mathcal{A}(\hat{Q}(t)) \tag{A.27}$$

Condition (A.27) implies that schedule $\tilde{\beta}$ only serves per-destination queues with queue backlogs larger than $\max(H_{max}, NR)$ throughout time interval $[(t-\tau)r_k, (t+\tau)r_k]$ and the queues it serves all have sufficient backlog to send. By (A.4), it must be a valid schedule under the O-BP policy throughout time interval $[(t-\tau)r_k, (t+\tau)r_k]$. This is because when (A.27) holds, it also implies that $Z_a^{(j)} > H_{max}$ for the corresponding output queue, as selected by OSPF. Therefore, for any $u_1 \leq u_2$, $u_1, u_2 \in [(t-\tau)r_k, (t+\tau)r_k]$ we have

$$T_\beta(u_2,\omega) - T_\beta(u_1,\omega) = 0$$

Similar to Theorem A.1.3.1, we can then prove that

$$\hat{T}_\beta(u_2) - \hat{T}_\beta(u_1) = 0$$

for any $u_1, u_2 \in [(t-\tau), (t+\tau)]$ with $u_1 \leq u_2$, from which (A.16) follows, proving the throughput optimality of O-BP.

The proof for the throughput optimality of E-BP is almost the same to that of O-BP except for

$$Q_a^{(c)}(s,\omega) > \max(M_a^{(c)} H_{max}, NR) \quad \text{for } (a,c) \in \mathcal{A}(\hat{Q}(t)) \tag{A.28}$$

and substituting (A.4) with (A.5) in the above analysis under Inequality (A.28), where $M_a^{(c)} \leq M_a$ is the subset of all M_a neighbors of node a that are on a shortest path for commodity c. The analysis holds because when (A.28) holds, it also implies that $Z_a^{(j)} > H_{max}$ for at least one of the $M_a^{(c)}$ output queues, even if the content of $Q_a^{(c)}$ is evenly distributed over all $M_a^{(c)}$ output queues in the worst-case. □

A.2 Proof for Chapter 4

In the appendix, to make it more readable, on the pruned graph G' with the node set $U' \cup \mathsf{N}'$, we use $+$ and $-$ to denote the set operations of union and minus respectively, use s and $\{s\}$ exchangeably for a set element s, and use $\pi(s/s')$ to denote $\pi(s+s') - \pi(s')$. Also, for any node $u \in U'$, denote $\mathsf{N}'(u)$ as the set of nodes in N' connecting to u.

Proof of Theorem 4.3.1. This theorem is a loose estimation based on the result proved in [11], where they show that to maximize the sum of a submodular function and a supermodular function, greedy algorithm can achieve $\frac{1}{\mathbb{c}_b}[1 - e^{-(1-\mathbb{c}^p)\mathbb{c}_b}]$ performance bound under the cardinality constraint, where $\mathbb{c}_b$ and $\mathbb{c}^p$ are the submodular curvature and

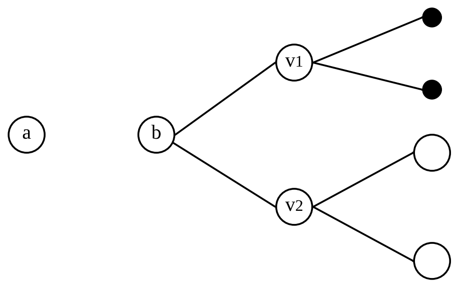


Figure A.1: A testimony of Proposition 4.4.1

supermodular curvature respectively. Since $\mathbb{c}_b \leq 1$, $\frac{1}{\mathbb{c}_b}[1 - e^{-(1-\mathbb{c}^p)\mathbb{c}_b}] \geq [1 - e^{-(1-\mathbb{c}^p)}]$. Also, when f is purely monotone supermodular, then greedy yields $(1 - \mathbb{c}^p)$ directly according to [11]. □

Proof of Proposition 4.4.1. To see this, let us check Figure A.1 where the black nodes are negative. If we set $X = \{a\}$ and $Y = \{a, b\}$, then it can be verified that $\pi(v_1/X) \leq \pi(X)$ and $\pi(v_2/X) \geq \pi(X)$ which means π is non-monotone, $\pi(v_1/X) \leq \pi(v_1/Y)$ which means π is non-submodular, and $\pi(v_2/X) \geq \pi(v_2/Y)$ which means π is non-supermodular. □

Proof of Proposition 4.5.1. We need to prove for $X \subseteq Y$ and $v \in U' - Y$, $\pi(v/X) \leq \pi(v/Y)$. It can be shown that on the pruned graph, $\pi(v/X) = w_v + |\mathsf{N}'(X)| - |\mathsf{N}'(v + X)|$ $= w_v - |\mathsf{N}'(v) - \mathsf{N}'(X)|$. Therefore, $\pi(v/X) = w_v - |\mathsf{N}'(v) - \mathsf{N}'(X)| \leq w_v - |\mathsf{N}'(v) - \mathsf{N}'(Y)| = \pi(v/Y)$, since $X \subseteq Y \Rightarrow \mathsf{N}'(X) \subseteq \mathsf{N}'(Y) \Rightarrow \mathsf{N}'(v) - \mathsf{N}'(X) \supseteq \mathsf{N}'(v) - \mathsf{N}'(X) \Rightarrow |\mathsf{N}'(v) - \mathsf{N}'(X)|$ $\geq |\mathsf{N}'(v) - \mathsf{N}'(Y)|$. □

Proof of Theorem 4.4.1. Consider two bipartite graphs G and G'. $G = (U, V, E)$, where $U = \{u_1, \cdots, u_k, u_{k+1}, \cdots, u_{2k}\}$, $V = \{v_1, \cdots, v_k\}$. For $1 \leq i \leq k$ and $2 \leq j \leq k$, we link u_i and v_j with an edge, and then for $(k + 1) \leq i \leq 2k$ and $1 \leq j \leq k$, we link an edge between u_i and v_j. G' has the same node sets U and V as G has, however, G' has a different edge set: for $1 \leq i \leq 2k$ and $1 \leq j \leq k$, we link u_i with v_j. For both G and G', let $\mathsf{N} = V$.

Let $R = \{u_1, \cdots, u_k\}$. If we think of *SPE* on G and G', and denote $\pi(S)$ and $\pi'(S)$ the benefits of a set S on G and G' respectively, it can be seen that $\pi(R) = 1$, $\pi'(R) =$

0, and $\pi(S) = \pi'(S) = 0$ if and only if $S \neq R$. Next, we adopt a proof technique used in [75]. For any algorithm that maximizes $\pi(S)$, all function evaluations are the same as it maximizes $\pi'(S)$, and $\pi'(S)$ is permutation symmetric. This means, the only way that an algorithm finds R is the random search. For any polynomial algorithm that searches $O(n^b)$ sets of size k, the probability of this algorithm to find R is $\frac{O(n^b)}{\binom{n}{k}}$. Since $\binom{n}{k} \geq (\frac{n}{k})^k$, $k = \frac{n}{3}$, and $n^b = O(3^{\epsilon n})$ for any small $\epsilon > 0$, this probability is $\frac{O(n^b)}{\binom{n}{n/3}} \leq \frac{O(n^b)}{3^{n/3}} = O\left(3^{-(1/3-\epsilon)n}\right)$. Therefore, for any polynomial algorithm, the probability that it finds R for G and returns the maximum payoff π as 1, or has any positive guarantee, is almost 0. $\square$

Proof of Theorem 4.5.1. We define an order of any k-size set S based on $\pi(S)$ in (4.3). Let $s_0 = \emptyset$. The order is $\{s_1, s_2, \cdots, s_k\}$, where

$$s_i = \underset{s \in S - \sum_{j=0}^{i-1} s_j}{\arg\max} \, \pi\left(s / \sum_{j=0}^{i-1} s_j\right).$$

In other words, $s_1 = \underset{s \in S}{\arg\max}\, \pi(s)$, $s_2 = \underset{s \in S - s_1}{\arg\max}\, \pi(s/s_1)$, and so on. This order is inspired on the steps when we generate a k-size set solution using GreedMax. Suppose under this order, $\hat{S} = \{\hat{s}_1, \cdots, \hat{s}_k\}$ and $S^* = \{s_1^*, \cdots, s_k^*\}$. Since $\hat{S}$ is obtained by GreedMax, we have $\pi(\hat{s}_1) \geq \pi(s_1^*)$.

We then show $\pi(\hat{s}_1 + \hat{s}_2) \geq \pi(s_1^* + s_2^*) - 1$. First, because of the way GreedMax works, we must have $\pi(\hat{s_2}/\hat{s}_1) \geq \pi(s_2^*/\hat{s}_1)$. Second, for any two sets x and y, $\pi(x/y)$ $= w_x + |\mathsf{N}'(y)| - |\mathsf{N}'(x+y)| = w_x - |\mathsf{N}'(x) - \mathsf{N}'(y)|$. Therefore, $\pi(s_2^*/\hat{s}_1) - \pi(s_2^*/s_1^*) =$ $|\mathsf{N}'(s_2^*) - \mathsf{N}'(s_1^*)| - |\mathsf{N}'(s_2^*) - \mathsf{N}'(\hat{s}_1)| \geq |\mathsf{N}'(s_2^*) - \mathsf{N}'(s_1^*)| - |\mathsf{N}'(s_2^*)| = -|\mathsf{N}'(s_2^*) \cap \mathsf{N}'(s_1^*)|$. Third, we will show that $|\mathsf{N}'(s_2^*) \cap \mathsf{N}'(s_1^*)| \leq 1$. Otherwise, we will have $|\mathsf{N}'(s_2^*) \cap \mathsf{N}'(s_1^*)| \geq 2$. Suppose v_1 and v_2 are two nodes in $\mathsf{N}'(s_2^*) \cap \mathsf{N}'(s_1^*)$, then $< s_1^*, v_1, s_2^*, v_2 >$ forms a cycle, which contradicts with our assumption that G' is a forest. Since $|\mathsf{N}'(s_2^*) \cap \mathsf{N}'(s_1^*)| \leq 1$ is true, we have $\pi(s_2^*/\hat{s}_1) - \pi(s_2^*/s_1^*) \geq -1$.

Similar to the proof above, we can further show that $\pi(\hat{s}_1 + \hat{s}_2 + \hat{s}_3) \geq \pi(s_1^* + s_2^* + s_3^*)$

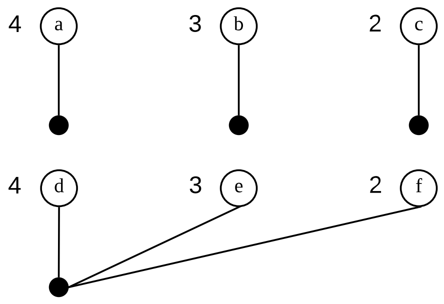


Figure A.2: An example with tight bound of Theorem 4.5.1

$-$ 2. In general, we can prove for any $1 \leq i \leq k$, it must be true that $\pi\left(\sum_{j=1}^{i} \hat{s}_j\right) \geq \pi\left(\sum_{j=1}^{i} s_j^*\right) - (i-1)$. Therefore we have proved $\pi(\hat{S}) = \pi\left(\sum_{j=1}^{k} \hat{s}_j\right) \geq \pi\left(\sum_{j=1}^{k} s_j^*\right) - (k-1) = \pi(S^*) - (k-1)$.

This bound is tight. Check Figure A.2 where the black nodes are the negative nodes and the numbers left to the circles are the node weights. When $k = 3$, the GreedMax may pick $\{a, b, c\}$ as the seed set and the benefit is 6, however the optimal seed set is $\{d, e, f\}$ whose benefit is 8. □

Proof of Theorem 4.5.2. For any set S and an arbitrary node $u_i \in U'$, $\pi(u_i/S) = w_{u_i} - |\mathsf{N}'(u_i) - \mathsf{N}'(S)| \geq w_{u_i} - |\mathsf{N}'(u_i)| \geq 0$ since $\forall u_i \in U'$. This means the benefit $\pi(S)$ is monotone increasing. According to proposition 4.5.1 and Theorem 4.3.1, *SPE* becomes a maximization problem of a monotone increasing supermodular problem, and GreedMax makes $(1 - \mathbb{e}^p)$ guarantee. □

Proof of Lemma 4.6.1:

$$\begin{aligned} F_{\mathcal{H}}(S) &= F_{\mathcal{H}^+}(S) - F_{\mathcal{H}^-}(S) \\ &= \sum_{v \in V \setminus \mathsf{N}'} \frac{1}{n} \cdot \Pr[S \text{ and } v\text{'s RR set intersect}] \\ &\quad - \sum_{v \in \mathsf{N}'} \frac{1}{n} \cdot \Pr[S \text{ and } v\text{'s RR set intersect}] \end{aligned}$$

$$\begin{aligned} &= \frac{1}{n}\left[\sum_{v\in V\setminus \mathsf{N}'} p(v) - \sum_{v\in \mathsf{N}'} p(v)\right] \\ &= \frac{1}{n}\left[\pi^+(S) - \pi^-(S)\right] = \frac{\pi(S)}{n}. \qquad \blacksquare \end{aligned}$$

For a superset $\mathcal{H} = \{H_1, H_2, \cdots, H_\rho\}$ and a seed set S, we define a random variable $x_i \in \{+1, 0, -1\}, 1 \le i \le \rho$. If $S \cap H_i = \emptyset$ then $x_i = 0$, else if $S \cap H_i \neq \emptyset$ and $H_i \in \mathcal{H}^+$ then $x_i = +1$, else if $S \cap H_i \neq \emptyset$ and $H_i \in \mathcal{H}^-$ then $x_i = -1$.

Let $a = \pi(S)/n$, then

$$a = \mathbb{E}\left[F_{\mathcal{H}}(S)\right]. \tag{A.29}$$

We have following corollaries,

Corollary A.2.1. *For $\epsilon > 0$*

$$\Pr\left[\sum_{i=1}^{\theta} x_i - \theta a \ge \epsilon \cdot \theta a\right] \le \exp\left(-\frac{\epsilon^2}{2 + \frac{2}{3}\epsilon} \cdot \theta a\right).$$

Corollary A.2.2. *For $\epsilon > 0$*

$$\Pr\left[\sum_{i=1}^{\theta} x_i - \theta a \le -\epsilon \cdot \theta a\right] \le \exp\left(-\frac{\epsilon^2}{2} \cdot \theta a\right).$$

Proof of Lemma 4.6.2. First we prove $n \cdot F_{\mathcal{H}}(S) \le (1+\epsilon) \cdot \pi(S)$ is true with probability at least $1 - \delta$. This is because

$$\begin{aligned} &\Pr\left[n \cdot F_{\mathcal{H}}(S) \ge (1+\epsilon) \cdot \pi(S)\right] \\ =\ &\Pr\left[F_{\mathcal{H}}(S) - a \ge \epsilon a\right] \\ =\ &\Pr\left[\sum_{i=1}^{\rho'} x_i - \rho' a \ge \epsilon \cdot \rho' a\right] \end{aligned}$$

$$
\begin{aligned}
&\le \exp\left(-\frac{\epsilon^2}{2+\frac{2}{3}\epsilon}\cdot \rho' a\right)\\
&\le \exp\left(-\frac{\epsilon^2}{2}\cdot \rho' \cdot \frac{\pi(S)}{n}\right)\\
&\le \exp\left(-\frac{\epsilon^2}{2}\cdot \frac{2n\cdot \ln(1/\delta)}{\epsilon^2}\cdot \frac{1}{n}\right) \le \delta
\end{aligned}
$$

Similarly, we can prove $n \cdot F_{\mathcal{H}}(S) \ge (1-\epsilon)\cdot \pi(S)$ stands with probability at least $1-\delta$. $\square$

Proof of Lemma 4.6.3. It is not hard to prove both $F_{\mathcal{H}^+}(S)$ and $F_{\mathcal{H}^-}(S)$ are submodular w.r.t. S, therefore $F_{\mathcal{H}}(S) = F_{\mathcal{H}^+}(S) - F_{\mathcal{H}^-}(S)$ is the summation of a submodular function and a supermodular function. By Theorem 4.3.1, the greedy algorithm gives a solution $\hat{S}$ such that $F_{\mathcal{H}}(\hat{S}) \ge [1-e^{-(1-\mathbb{c}^p)}]\ F_{\mathcal{H}}(S^*_{\mathcal{H}})$, where $S^*_{\mathcal{H}}$ is the set that maximizes $F_{\mathcal{H}}$. By lemma 4.6.2, $(1+\epsilon)\cdot\pi(\hat{S}) \ge F_{\mathcal{H}}(\hat{S})$ with probability $(1-\delta/2)$, and $F_{\mathcal{H}}(S^*) \ge (1-\epsilon)\cdot\pi(S^*)$ with probability $1-\delta/2$. Conditioning on these two events, $\pi(\hat{S}) \ge \frac{1}{1+\epsilon}\cdot F_{\mathcal{H}}(\hat{S}) \ge \frac{1}{1+\epsilon}\cdot [1-e^{-(1-\mathbb{c}^p)}]\ F_{\mathcal{H}}(S^*_{\mathcal{H}}) \ge \frac{1}{1+\epsilon}\cdot [1-e^{-(1-\mathbb{c}^p)}]\ F_{\mathcal{H}}(S^*) \ge \frac{1-\epsilon}{1+\epsilon}\cdot [1-e^{-(1-\mathbb{c}^p)}]\ F_{\mathcal{H}}(S^*)$ with probability $1-\delta$. $\square$

Proof of Theorem 4.6.1. The first half of the theorem is the performance bound, and it directly comes from Lemma 4.6.3.

In the second half, we prove the time complexity. Denote AVG the expected number of edges checked by Alg. 3 to generate a RR set, obviously $AVG \le m$. Since Alg. 4 generates ρ RR sets, and the total complexity is $\rho \cdot AVG \le \rho\cdot m = \frac{2n\cdot(\ln 2+\ln(1/\delta))}{\epsilon^2}\cdot m = O\left(\ln(1/\delta)\cdot nm\cdot\epsilon^{-2}\right)$. The proof is complete. $\square$

In order to approximate the $\pi(S)$ of a given k-size set S, Alg. 5 - *Benefit Approximation* (BA) is devised. It can estimate any benefit π of a given k-size set S.

Algorithm 5: BA (Benefit Approximation)

```
input: $G, \mathsf{N},$ and a node set $S$;
output: a real number which is the estimated benefit of $S$.
1: $\mathcal{H} = \emptyset$, $\Pi \leftarrow 1$;
2: for i from 1 to $\lg n - 1$ do
3:     $x \leftarrow n/2^i$;
4:     $\lambda' = (2 + \frac{2}{3}\epsilon) \cdot \left(\ln \frac{1}{\delta} + \ln \lg n\right) \cdot \frac{n}{\epsilon^2}$;
5:     $\theta_i \leftarrow \lambda'/x$;
6:     while $|\mathcal{H}| \leq \theta_i$ do
7:         $\mathcal{H} \leftarrow \mathcal{H} \cup \mathrm{RRS}(G)$;
8:     end while;
9:     if $n \cdot F_{\mathcal{H}}(S) \geq (1+\epsilon) \cdot x$ then
10:        $\Pi \leftarrow n \cdot F_{\mathcal{H}}(S)/(1+\epsilon)$;
11:        break;
12:    end if
13: end for
14: return $\Pi$.
```

Bibliography

[1] Cisco IOS netflow. `http://www.cisco.com/warp/public/732/Tech/netflow/`.

[2] Abebe, R., Adamic, L. A., and Kleinberg, J. M. Mitigating overexposure in viral marketing. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018* (2018).

[3] Aizen, J., Huttenlocher, D., Kleinberg, J., and Novak, A. Traffic-based feedback on the web. *Proceedings of the National Academy of Sciences 101*, suppl 1 (2004), 5254–5260.

[4] Allalouf, M., and Shavitt, Y. Centralized and distributed approximation algorithms for routing and weighted max-min fair bandwidth allocation. In *IEEE Workshop on High Performance Switching and Routing* (2005), pp. 306–311.

[5] Alresaini, M., Sathiamoorthy, M., Krishnamachari, B., and Neely, M. J. Backpressure with adaptive redundancy (bwar). In *2012 Proceedings IEEE INFOCOM* (March 2012), pp. 2300–2308.

[6] Alresaini, M., Wright, K. L., Krishnamachari, B., and Neely, M. J. Backpressure delay enhancement for encounter-based mobile networks while sustaining throughput optimality. *IEEE/ACM Transactions on Networking 24*, 2 (April 2016), 1196–1208.

[7] ApS, M. 2017.

[8] Arthur, D., Motwani, R., Sharma, A., and Xu, Y. Pricing strategies for viral marketing on social networks. In *Internet and Network Economics* (Berlin, Heidelberg, 2009), S. Leonardi, Ed., Springer Berlin Heidelberg, pp. 101–112.

[9] Athanasopoulou, E., Bui, L. X., Ji, T., Srikant, R., and Stolyar, A. Back-pressure-based packet-by-packet adaptive routing in communication networks. *IEEE/ACM Transactions on Networking (TON) 21*, 1 (2013), 244–257.

[10] Awerbuch, B., and Leighton, T. A simple local-control approximation algorithm for multicommodity flow. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science* (Nov 1993), pp. 459–468.

[11] Bai, W., and Bilmes, J. Greed is still good: Maximizing monotone submodular+supermodular functions. *arXiv:1801.07413 [cs.DM]* (2018).

[12] Banerjee, A., Drake, J., Lang, J. P., Turner, B., Kompella, K., and Rekhter, Y. Generalized multiprotocol label switching: An overview of routing and management enhancements. *IEEE Communications Magazine 39*, 1 (2001), 144–150.

[13] Banerjee, D., and Mukherjee, B. Wavelength-routed optical networks: Linear formulation, resource budgeting tradeoffs, and a reconfiguration study. *IEEE/ACM Transactions on Networking 8*, 5 (2000), 598–607.

[14] Berger, J., and Heath, C. Where consumers diverge from others: Identity signaling and product domains. *Journal of Consumer Research 34*, 2 (2007), 121–134.

[15] Borgs, C., Brautbar, M., Chayes, J., and Lucier, B. Maximizing social influence in nearly optimal time. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms* (Philadelphia, PA, USA, 2014), SODA '14, Society for Industrial and Applied Mathematics, pp. 946–957.

[16] Boyd, S., and Vandenberghe, L. *Convex Optimization*, 1st ed. Cambridge University Press, March 2004.

[17] Bui, L., Srikant, R., and Stolyar, A. Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing. In *INFOCOM 2009, IEEE* (2009), IEEE, pp. 2936–2940.

[18] Byers, J. W., Mitzenmacher, M., and Zervas, G. The groupon effect on yelp ratings: A root cause analysis. In *Proceedings of the 13th ACM Conference on Electronic Commerce* (New York, NY, USA, 2012), EC '12, ACM, pp. 248–265.

[19] Cao, Z., and Zegura, E. W. Utility max-min: An application-oriented bandwidth allocation scheme. In *IEEE INFOCOM* (1999), pp. 793–801.

[20] Chen, W., Collins, A., Cummings, R., Ke, T., Liu, Z., Rincon, D., Sun, X., Wang, Y., Wei, W., and Yuan, Y. Influence maximization in social networks when negative opinions may emerge and propagate. In *Proceedings of the 2011 SIAM International Conference on Data Mining (SDM 2011)* (2011), pp. 379–390.

[21] Chen, W., Yuan, Y., and Zhang, L. Scalable influence maximization in social networks under the linear threshold model. ICDM '10.

[22] Chou, J., and Lin, B. Coarse optical circuit switching by default, rerouting over circuits for adaptation. *Journal of Optical Networking 8*, 1 (2009), 33–50.

[23] Chou, J., and Lin, B. Optimal multi-path routing and bandwidth allocation under utility max-min fairness. In *IEEE International Workshop on Quality of Service* (2009), pp. 1–9.

[24] Comellas, J., Martnez, R., Prat, J., Sales, V., and Junyent, G. Integrated IP/WDM routing in GMPLS-based optical networks. *IEEE Network 17*, 2 (2003), 22–27.

[25] Cui, Y., Yeh, E. M., and Liu, R. Enhancing the delay performance of dynamic backpressure algorithms. *IEEE/ACM Transactions on Networking 24*, 2 (April 2016), 954–967.

[26] Dai, J., and Prabhakar, B. The throughput of data switches with and without speedup. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (2000), vol. 2, IEEE, pp. 556–564.

[27] Dai, J. G., and Lin, W. Maximum pressure policies in stochastic processing networks. *Operations Research 53*, 2 (2005), 197–218.

[28] Dai, J. G., and Weiss, G. Stability and instability of fluid models for reentrant lines. *Mathematics of Operations Research 21*, 1 (1996), 115–134.

[29] Daliri Khomami, M., Rezvanian, A., Bagherpour, N., and Meybodi, M. Minimum positive influence dominating set and its application in influence maximization: a learning automata approach. *Applied Intelligence 48*, 3 (2018), 570–593.

[30] Das, S., Parulkar, G., and McKeown, N. Why OpenFlow/SDN can succeed where GMPLS failed. In *European Conference and Exhibition on Optical Communication* (2012).

[31] Das, S., Parulkar, G., and McKeown, N. Rethinking IP core networks. *Journal of Optical Communications and Networking 5*, 12 (2013), 1431–1442.

[32] Diamond, S., and Boyd, S. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research 17*, 83 (2016), 1–5.

[33] Eryilmaz, A., and Srikant, R. Joint congestion control, routing, and mac for stability and fairness in wireless networks. *IEEE Journal on Selected Areas in Communications 24*, 8 (2006), 1514–1524.

[34] Georgiadis, L., Neely, M. J., and Tassiulas, L. Resource allocation and cross-layer control in wireless networks. *Foundations and Trends in Networking 1*, 1 (2006), 1–144.

[35] Goyal, A., Lu, W., and Lakshmanan, L. Celf++: Optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th International Conference Companion on World Wide Web* (2011), WWW '11, ACM, pp. 47–48.

[36] Hartline, J., Mirrokni, V., and Sundararajan, M. Optimal marketing strategies over social networks. In *Proceedings of the 17th International Conference on World Wide Web* (2008), WWW '08, ACM, pp. 189–198.

[37] Hopps, C. Analysis of an equal-cost multi-path algorithm. `http://tools.ietf.org/html/rfc2992/`.

[38] Hu, Y., and Bulte, C. V. Nonmonotonic status effects in new product adoption. *Marketing Science 33*, 4 (2014), 509–533.

[39] Huelsermann, R., Gunkel, M., Meusburger, C., and Schupke, D. A. Cost modeling and evaluation of capital expenditures in optical multilayer networks. *Journal of Optical Networking 7*, 9 (2008), 814–833.

[40] Internet2. Advanced networking for leading-edge research and education. `http://www.internet2.edu/`.

[41] Iyer, S., Kompella, R. R., and McKeown, N. Designing packet buffers for router linecards. *IEEE/ACM Transactions on Networking (TON) 16*, 3 (2008), 705–717.

[42] Ji, B., Joo, C., and Shroff, N. B. Delay-based back-pressure scheduling in multihop wireless networks. *IEEE/ACM Trans. Netw. 21*, 5 (Oct. 2013), 1539–1552.

[43] Ji, B., Joo, C., and Shroff, N. B. Throughput-optimal scheduling in multihop wireless networks without per-flow information. *IEEE/ACM Trans. Netw. 21*, 2 (Apr. 2013), 634–647.

[44] Joshi, Y. V., Reibstein, D. J., and Zhang, Z. J. Turf wars: Product line strategies in competitive markets. *Marketing Science 35*, 1 (2016), 128–141.

[45] J.Tang, Zhang, R., Yao, Y., Zhao, Z., P.Wang, Li, H., and Yuan, J. Maximizing the spread of influence via the collective intelligence of discrete bat algorithm. *Knowledge-Based Systems 160* (2018), 88 – 103.

[46] Katz, M. L., and Shapiro, C. Network externalities, competition, and compatibility. *The American Economic Review 75*, 3 (1985), 424–440.

[47] Kempe, D., Kleinberg, J., and Tardos, E. Maximizing the spread of influence through a social network. KDD '03, ACM, pp. 137–146.

[48] Kovcs, B., and Sharkey, A. J. The paradox of publicity: How awards can negatively affect the evaluation of quality. *Administrative Science Quarterly 59*, 1 (2014), 1–33.

[49] Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE 103*, 1 (2015), 14–76.

[50] Lappas, T., Terzi, E., Gunopulos, D., and Mannila, H. Finding effectors in social networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2010), KDD '10, ACM, pp. 1059–1068.

[51] Leskovec, J., Kleinberg, J., and Faloutsos, C. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data 1*, 1 (Mar. 2007).

[52] Li, G., Wang, D., Yates, J., Doverspike, R., and Kalmanek, C. IP over optical cross-connect architectures. *IEEE Communications Magazine 45*, 2 (2007), 34–39.

[53] Lu, W., and Lakshmanan, L. Profit maximization over social networks. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining* (Washington, DC, USA, 2012), ICDM '12, IEEE Computer Society, pp. 479–488.

[54] Magnani, A., and Boyd, S. Convex piecewise-linear fitting. *Optimization and Engineering 10*, 1 (2009), 1–17.

[55] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review 39*, 2 (2008), 69–74.

[56] Medina, A., Taft, N., Salamatian, K., Bhattacharyya, S., and Diot, C. Traffic matrix estimation: Existing techniques and new directions. *ACM SIGCOMM Computer Communication Review 32*, 4 (2002), 161–174.

[57] Mo, J., and Walrand, J. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking 8*, 5 (October 2000), 556–567.

[58] Morrow, M. J., Tatipamula, M., and Farrel, A. GMPLS: The promise of the next-generation optical control plane. *IEEE Communications Magazine 43*, 7 (2005), 26–27.

[59] Moy, J. OSPF version 2. `http://tools.ietf.org/html/rfc2328/`.

[60] Moy, J. OSPF version 2. `www.ieft.org/rfc/rfc2328.txt`, March 1994.

[61] Neely, M. J., Modiano, E., and Li, C. Fairness and optimal stochastic control for heterogeneous networks. In *Proceedings of IEEE INFOCOM* (2005), IEEE Press.

[62] Neely, M. J., Modiano, E., and Rohrs, C. E. Dynamic power allocation and routing for time-varying wireless networks. *IEEE Journal on Selected Areas in Communications 23*, 1 (2005), 89–103.

[63] Pham, C., Duong, H., Bui, B., and Thai, M. Budgeted competitive influence maximization on online social networks. In *Computational Data and Social Networks* (Cham, 2018), Springer International Publishing, pp. 13–24.

[64] Radunovic, B., and Boudec, J.-Y. L. A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Transactions on Networking 15*, 5 (2007), 1073–1083.

[65] Ramamurthy, B., and Ramakrishnan, A. Virtual topology reconfiguration of wavelength-routed optical WDM networks. In *IEEE Global Telecommunications Conference* (2000), pp. 1269–1275.

[66] Richardson, M., Agrawal, R., and Domingos, P. Trust management for the semantic web. In *Proceedings of the Second International Conference on Semantic Web Conference* (2003), LNCS-ISWC'03, Springer-Verlag, pp. 351–368.

[67] Roughan, M., Greenberg, A., Kalmanek, C., Rumsewicz, M., Yates, J., and Zhang, Y. Experience in measuring internet backbone traffic variability: Models, metrics, measurements and meaning. In *International Teletraffic Congress (ITC)* (2003), pp. 379–388.

[68] Rubenstein, D., Kurose, J., and Towsley, D. The impact of multicast layering on network fairness. *EEE/ACM Transactions on Networking 10*, 2 (2002), 169–182.

[69] Ryu, J., Ying, L., and Shakkottai, S. Back-pressure routing for intermittently connected networks. In *Proceedings of the 29th Conference on Information Communications* (Piscataway, NJ, USA, 2010), INFOCOM'10, IEEE Press, pp. 306–310.

[70] Seferoglu, H., and Modiano, E. Separation of routing and scheduling in backpressure-based wireless networks. *IEEE/ACM Transactions on Networking 24*, 3 (June 2016), 1787–1800.

[71] Sengupta, S., Kumar, V., and Saha, D. Switched optical backbone for cost-effective scalable core IP networks. *IEEE Communications Magazine 41*, 6 (2003), 60–70.

[72] Shenker, S., Casado, M., Koponen, T., and McKeown, N. The future of networking, and the past of protocols. In *Open Networking Summit* (2011), pp. 1–30.

[73] Shrimali, G., and McKeown, N. Building packet buffers using interleaved memories. In *Proceedings of the Workshop on High Performance Switching and Routing (HPSR)* (May 2005), IEEE Press.

[74] Simmons, J. *Optical Network Design and Planning.* Springer, 2014.

[75] Svitkina, Z., and Fleischer, L. Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM J. Comput. 40*, 6 (Dec. 2011), 1715–1737.

[76] Tang, Y., Shi, Y., and Xiao, X. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2015), SIGMOD '15, ACM, pp. 1539–1554.

[77] Tang, Y., Xiao, X., and Shi, Y. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data* (2014), SIGMOD '14, pp. 75–86.

[78] Tassiulas, L., and Ephremides, A. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control 37*, 12 (Dec 1992), 1936–1948.

[79] Tornatore, M., Maier, G., and Pattavina, A. WDM network optimization by ILP based on source formulation. In *IEEE INFOCOM* (2002), pp. 1813–1821.

[80] Wang, H., and Lin, B. Block-based packet buffer with deterministic packet departures. In *Proceedings of the 11th International Conference on High Performance Switching (HPSR)* (June 2010), IEEE Press.

[81] Wang, H., Zhao, H., Lin, B., and Xu, J. Robust pipelined memory system with worst case performance guarantee for network processing. *IEEE/ACM Transactions on Computers (TC) 61*, 10 (2012), 1386–1400.

[82] Wen, S., Haghighi, M. S., Chen, C., Xiang, Y., Zhou, W., and Jia, W. A sword with two edges: Propagation studies on both positive and negative information in online social networks. *IEEE Transactions on Computers 64*, 3 (March 2015), 640–653.

[83] Xia, W., Wen, Y., Foh, C. H., Niyato, D., and Xie, H. A survey on software-defined networking. *IEEE Communications Surveys and Tutorials 17*, 1 (2015), 27–51.

[84] Yang, J., and Leskovec, J. Defining and evaluating network communities based on ground-truth. *Knowl. Inf. Syst. 42*, 1 (Jan. 2015), 181–213.

[85] Yin, P., Yang, S., Xu, J., Dai, J., and Lin, B. Improving backpressure-based adaptive routing via incremental expansion of routing choices. In *Proceedings of the Symposium on Architectures for Networking and Communications Systems* (2017), pp. 1–12.

[86] Ying, L., Shakkottai, S., Reddy, A., and Liu, S. On combining shortest-path and back-pressure routing over multihop wireless networks. *IEEE/ACM Transactions on Networking (TON) 19*, 3 (2011), 841–854.

[87] Ying, L., Srikant, R., and Towsley, D. Cluster-based back-pressure routing algorithm. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE* (2008), IEEE, pp. 484–492.

[88] Yoo, S. B. Optical-label switching, MPLS, MPLambdaS, and GMPLS. *Optical Networking Magazine 4*, 3 (2003), 17–31.

[89] Zhang, H., Dinh, T. N., and Thai, M. T. Maximizing the spread of positive influence in online social networks. In *2013 IEEE 33rd International Conference on Distributed Computing Systems* (2013), pp. 317–326.

[90] Zhang, Y. Abilene traffic matrices. `http://www.cs.utexas.edu/~yzhang/research/AbileneTM/`.

[91] Zhang, Z., Shi, Y., Willson, J., Du, D. Z., and Tong, G. Viral marketing with positive influence. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications* (2017), pp. 1–8.

[92] Zhu, Y., Li, D., Yan, R., Wu, W., and Bi, Y. Maximizing the influence and profit in social networks. *IEEE Transactions on Computational Social Systems PP*, 99 (2017), 1–11.