1-1-2014

Fast, Distributed Optimization Strategies for Resource Allocation in Networks

Michael Zargham
University of Pennsylvania, zargham@seas.upenn.edu

Fast, Distributed Optimization Strategies for Resource Allocation in Networks

Abstract
Many challenges in network science and engineering today arise from systems composed of many individual agents interacting over a network. Such problems range from humans interacting with each other in social networks to computers processing and exchanging information over wired or wireless networks. In any application where information is spread out spatially, solutions must address information aggregation in addition to the decision process itself. Intelligently addressing the trade off between information aggregation and decision accuracy is fundamental to finding solutions quickly and accurately. Network optimization challenges such as these have generated a lot of interest in distributed optimization methods. The field of distributed optimization deals with iterative methods which perform calculations using locally available information. Early methods such as subgradient descent suffer very slow convergence rates because the underlying optimization method is a first order method. My work addresses problems in the area of network optimization and control with an emphasis on accelerating the rate of convergence by using a faster underlying optimization method. In the case of convex network flow optimization, the problem is transformed to the dual domain, moving the equality constraints which guarantee flow conservation into the objective. The Newton direction can be computed locally by using a consensus iteration to solve a Poisson equation, but this requires a lot of communication between neighboring nodes. Accelerated Dual Descent (ADD) is an approximate Newton method, which significantly reduces the communication requirement. Defining a stochastic version of the convex network flow problem with edge capacities yields a problem equivalent to the queue stability problem studied in the backpressure literature. Accelerated Backpressure (ABP) is developed to solve the queue stabilization problem. A queue reduction method is introduced by merging ideas from integral control and momentum based optimization.

Degree Type
Dissertation

Degree Name
Doctor of Philosophy (PhD)

Graduate Group
Electrical & Systems Engineering

First Advisor
Ali Jadbabaie

Second Advisor
Alejandro Ribeiro

Keywords
Control Theory, Distributed Algorithms, Network Science, Optimization

FAST, DISTRIBUTED OPTIMIZATION STRATEGIES FOR RESOURCE ALLOCATION IN NETWORKS

Michael C. Zargham

A DISSERTATION

in

Electrical and Systems Engineering

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Doctor of Philosophy

2014

ACKNOWLEDGMENT

I would like to thank my parents for decades of guidance and my beautiful wife Stacey for her constant loving support.

Over my academic career, Victor and Alejandro have been amazing resources, serving as teachers, mentors and collaborators. Special thanks to Aryan for his deep insights and recent contributions to my research. Thanks to Haitham and Rasul for their help editing my thesis and last but not least, thank you to my advisor, Ali.

ABSTRACT

FAST, DISTRIBUTED OPTIMIZATION STRATEGIES FOR RESOURCE ALLOCATION IN NETWORKS

Michael C. Zargham

Ali Jadbabaie

Many challenges in network science and engineering today arise from systems composed of many individual agents interacting over a network. Such problems range from humans interacting with each other in social networks to computers processing and exchanging information over wired or wireless networks. In any application where information is spread out spatially, solutions must address information aggregation in addition to the decision process itself. Intelligently addressing the trade off between information aggregation and decision accuracy is fundamental to finding solutions quickly and accurately. Network optimization challenges such as these have generated a lot of interest in distributed optimization methods. The field of distributed optimization deals with iterative methods which perform calculations using locally available information. Early methods such as subgradient descent suffer very slow convergence rates because the underlying optimization method is a first order method. My work addresses problems in the area of network optimization and control with an emphasis on accelerating the rate of convergence by using a faster underlying optimization method. In the case of convex network flow optimization, the problem is transformed to the dual domain, moving the equality constraints which guarantee flow conservation into the objective. The Newton direction can be computed locally by using a consensus iteration to solve a Poisson equation, but this requires a lot of communication between neighboring nodes. Accelerated Dual Descent (ADD) is an approximate Newton method, which significantly reduces the communication requirement. Defining a stochastic version of the convex network flow problem with edge capacities yields a problem equivalent to the queue stability problem studied in the backpressure literature. Accelerated Backpressure (ABP) is developed to solve the queue stabilization problem. A queue reduction method is introduced by merging ideas from integral control and momentum based optimization.

Contents

Chapter 1

Introduction

The convex network flow optimization is a problem of getting commodities from arrival points to destinations by routing them through a network – a graph composed of nodes and links between nodes. The problem is characterized by convex functions on the links which encode the cost to transport the commodity across those links and a set of constraints requiring that the commodities are conserved at the nodes. The objective is to determine the least costly way to route the commodities through the network while ensuring all commodities which enter the network reach their destination.

While applications vary widely, the transportation problem is the most intuitive way to visualize the convex network flow problem. Formulations of the problem vary in complexity and may include one or more unique commodities with their own arrival and destination nodes. Arrivals may be deterministic or stochastic and there may be local constraints on the edges restricting the units which may be transported across them at any time.

Convex optimization problems are a broad class of problems which can be solved in polynomial time. Due to the fundamentally distributed nature of the convex network flow optimization problem, it may be solved using protocols implemented locally at the nodes in the network. These distributed algorithms are relevant because in many routing applications nodes are responsible for choosing how to forward the commodities they recieve but are only aware of their immediate neighborhood–

the other nodes with which they have links.

In this work, a novel distributed algorithm for convex network flow optimization is proposed and applied to solve packet routing problems. In Section 1.1, the origins and applications of the convex network flow problem are outlined. In Section 1.2, the literature on algorithms for convex network flow optimization is reviewed and state of the art algorithms are critiqued for their strengths and weaknesses. In Section 1.3, the primary motivating application– packet routing problems are explored in depth and the connection between queue stabilization formulations and the network flow optimization problem is detailed.

1.1 Motivation for Network Flow Optimization

The convex network flow optimization problem is fundamental problem in the field of operations research as it can be used to model transportation as well as complex scheduling, manufacturing and staffing problems. Early work on the subject comes from the operations research literature and focused on linear objectives as in [Chen and Saigal, 1977]. Formal problem formulations involving convex costs for operations research applications can be found in [Goldberg et al., 1990]. Applications in electrical engineering and computer science are more recent, arriving as the control and networking communities have adopted powerful tools from convex optimization. Two textbooks facilitate this cross-over, [Rockafellar, 1984] formulates the convex network flow optimization problem and analyzes a variety of optimization based methods and [Bertsekas, 1998] formulates an array network optimization problems and convex optimization algorithms that can be used to solve them. Applications in recent research range from computer vision to control of smart power grids to the pack forwarding in wireless networks.

1.1.1 Operations Research

The most basic operations research application is the uncapacitated transshimpment problem which is one of many problem formulations detailed in [Goldberg et al., 1990]. The transshipment problem simply requires finding the least costly way to ship commodities from a source to a destination within a transportation network. Various generalizes including the additon of capacity constaints, multiple commodities and randomized commodity demand are relevant making transshipment a good problem for theoretical study. Further study of the transshipment problem can be found in [Herer and Tzur, 2001]; demand is dynamic and commodities can be stored. This version of the transshipment problem is comparable to the packet routing problems where queuing packets for transmission is part of the underlying model.

Scheduling, staffing and manufacturing problems are less obvious applications of the network flow optimization problem. These applications are presented alongside the transshipment problem as motivation for convex network flow problems in [Orlin, 1984]. This work focuses on the cyclic nature of these problems introducing an objective encoding the efficiency with which tasks are completed, leading to convex costs where earlier operations research had used primarily linear cost functions.

Recent research on complex manufacturing tasks maps the flow of materials and the equipment used for processing into a convex network flow optimization problem to solve a simultanous scheduling and routing problem. In [Shu-Cherng and Qi, 2003], the emphasis is on the problem formulation which allows the integration of various resource types. The model presented allows access to a wide array of convex optimization tools. In [Hindi and Ruml, 2006], the use of the convex network flow model of a manufacturing process is leveraged to handle re-entry of materials. Since manufacturing processes are traditionally one directional, the introduction of feedback in the form of material re-entry renders many common process optimzation schemes inapplicable. Network optimization methods, however, handle feedback easily.

1.1.2 Electrical Engineering and Computer Science

Power grids face increasing loads as well as heterogeous production modes and an increase in local power generation through solar and other immerging technologies. In [Adhikari et al., 2012], a network flow optimization problem is proposed to model power transmission control between transfer stations. More extensive, integrated infractructure models are also proposed to leverage multi-commodity versions of the network flow optimization problem. While this work assumes a linear objective, the convex optimization literature provides tools sufficient for consideration of models with more complex cost functions.

In computer vision, the dual form of the convex network flow optimization problem appears in panaramic image stitching and image restoration problems. In [Kolmogorov and Shioura, 2007], the authors outline these and other vision related applications and proceed to present an iterative primal dual algorithm to solve the problem. The algorithm presented is centralized; implementation of distributed protocols would allow the use of multiple processors reducing time complexity, which is a often the primary barrier in large scale vision tasks.

The most direct application of the convex network flow problem is packet routing. In [Wu et al., 2007], a robust packet routing problem for wireless networks where power allocated for packet transmissions across noisy communication channels determines expected effective transmission rates. As formulated, the robust routing problem is a convex quadratic cost network flow optmization problem. This particular application requires an solution method using local protocols, which until recently have been too slow to be practical. Multi-layer routing problems requiring simultaneous scheduling and routing also fit into the network flow optimization framework. This application is discussed extensively in Section 1.3.

1.2 Algorithms for Convex Network Flow Optimization

The convex network flow optimization problem has a fundamentally distributed nature because the costs are incurred on the edges of an underlying network and the constraints are enforced on the nodes. In reviewing the literature, algorithms are evaluated primarily on their speed of convergence and the degree to which they can be implemented using node level protocols. In order to exploit the distributed structure of the problem, the Lagrange dual is formed, introducing penalty variables for violations of the conservation constraints at each node. From duality theory, we can compute the optimal flows by solving for the optimal penalty variables via the Lagrange dual problem and reconstruct the optimal primal variables from that solution. Dualization leads to a variety of dual and primal-dual optimization methods with various degrees of decentralization.

1.2.1 Dual Gradient Methods

Dual subgradient descent is the most basic distributed algorithm. In the Lagrange dual problem, the conservation constraints enter the objective multiplied by their respective penalties. Due to the problem structure, the flow on any edge can be determined uniquely from the penalties at its boundary nodes. In convex network flow optimization, the dual subgradient is a unique gradient vector under reasonable assumptions on the primal objective. The dual optimal is computed by alternately updating the penalties by descending the dual gradient and updating the primal (flow) variables according to the current penalties. The resulting dual gradient descent method is fully distributed requiring local edge computations and local node observations.

Unfortunately, gradient descent is a very slow method; extensive analysis of basic distributed gradient methods can be found in [Nedić and Ozdaglar, 2009b] and [Lobel and Ozdaglar, 2011]. These results reflect recent effort to leverage the decentralization capabilities of the gradient methods, the slow nature of these methods has been established for decades. For earlier analysis, see [Shor, 1985].

Gradient methods rely on stepsizes in order to guarantee convergence. A major innovation in gradient based methods is the design of step size rules which accelerate the convergence rate. Polyak's step size introduced in [Polyak, 1967], uses a renormalization to ensure that the gradient step taken has the magnitude of the optimality gap. The downside of Polyak's stepsize is that it is not computable easily using local protocols due to a division by the norm of the dual gradient.

Another branch of stepsize accelerations are found in [Nesterov, 1983]. Nesterov's methods have two key features, the optimal selection of stepsize values based on strong convexity and Lipschitz coefficients and the introduction of a momentum term commonly referred to as the heavy ball method. In the case of the convex network flow problem, knowledge of global bounding constants such as Lipschitz or strong convexity coefficients are not assumed to be known therefore, Nesterov's stepsize cannot be implemented. Momementum based methods are relevant and are explored in this work within the context of packet routing problems.

Another way to characterize these stepsize accelerations is through augementing the statespace to include a momentum aggregating state and the original state. This notion is independently proposed in [Lessard et al., 2014] and [Zargham et al., 2014a]. In [Lessard et al., 2014], the author studies the stability of deterministic convex optimization algorithms, treating the gradient update as non-linear perturbation in an otherwise linear system and using decoupling techniques to analyze stability properties. In [Zargham et al., 2014a], the state augmentation arises naturally as a consequence of applying a momentum method (detailed in Chapter 5) to a queue stabilization problem formulated as a stochastic network flow optimization.

Stochastic gradient methods are used when only a noisy version of the gradient is observable. This is the case for the network flow optimization problem when the demand vectors are stochastic. The stochastic gradient descent (SGD) algorithm, is the benchmark method in stochastic optimization. As detailed in [Bertsekas et al., 2003], it exhibits a linear convergence rate. Convergence to the optimal point is achieved through the introduction of a decaying stepsize which is square summable but not summable. Stochastic average gradient (SAG), introduced in [Schmidt et al., 2013] demonstrates convergence to the optimal point with a fixed stepsize by descending in a direction which is the average of recent stochastic gradients. In practice, averaging stochastic gradients is similar to implementing momentum based stochastic gradient methods like those proposed in [Tseng, 1998] and [Flam, 2004].

1.2.2 Augmented Gradient Methods

Recent efforts to accelerate gradient based methods have focused on augementation of the objective to include addtional penalty functions. Changes in the objective become reflected in the gradient and in the ideal case steer the descent algorithm to the optimal solution to the original problem along a more direct path. Proximal methods augment the objective by applying a quadratic penalty on the update step. Fast Itertative Shrinking-Threshold Algorithm (FISTA), is the state of the art proximal method; first introduced in [Beck and Teboulle, 2009a], FISTA is formulated for distributed implementation in [Chen and Ozdaglar, 2012].

Proximal methods are shown to be particularly effective in the case of stochastic optimization problems in [Duchi et al., 2011]. This method improves convergence by adding weight to features which are observed less frequently. While not designed explicitly for a network flow optimization, this method is built on stochastic subgradient descent, so derivation of a distributed variant for the convex network flow problem is plausable. Such a generalization would be particularly relevant for packet routing applications where only stochastic gradients are observable.

Another augmented method which shows great promise in the distributed optimization literature is the Alternating Direction Method of Multipliers (ADMM). This method breaks the constraint into pieces so that the constraint penalties can be updated independently. The augmentation term in the objective is a coefficient times the norm of the constraint violation. This scaling coefficient dictates the optimal stepsize leading to a very efficient gradient descent path. Details on ADMM can be found in chapter 17 of [Bonnans et al., 2006]. Independently in [Wei and Ozdaglar, 2013] and [Ling and Ribeiro, 2013], ADMM is generalized for distributed implementation.

1.2.3 Newton-Type Methods

Faster convergence rates can generally be achieved moving to algorithms incorporating higher order information. Newton-type methods are those which use second order information in the form of a dual Hessian to adjust the descent direction to account for curvature. The foundational second order methods are Newton's method and conjugate gradient, the details can be found in [Bertsekas, 1999]. Neither can be applied directly in our case due to global operations: Newton's method requires the inversion of the dual Hessian and conjugate gradient requires a variety of global inner product computations.

An early Newton-type algorithm that leverages the distributed structure of the convex network flow optimization problem is outlined in [Bertsekas and Gafni, 1983]. An approximation of the Newton direction is computed by solving a linear equation iteratively. The iterative solver proposed is based on the conjugate gradient method, which in each iteration requires an inner product computation, preventing a completely distributed implementation. An asynchronous variation using a Gauss-Siedel relaxation is proposed in [Bertsekas and Baz, 1987]. Allowing nodes to take turns updating their penalty variables unitlaterally is shown to eventually lead to optimal penalty variables for all nodes. The main benefit of this method is its asynchronous and completely distributed implementation, however it suffers prohibitively slow convergence. Another approach proposed in [Klincewicz, 1983], computes the Newton direction via conjugate vectors. The method is implemented using spanning trees to compute the necessary inner products. This level of network wide node coordination is outside the scope of our current research as we do not assume the global network knowledge needed to build spanning trees.

In each case, the challenge in implementing a distributed Newton-type algorithm is entirely in the computation of the Newton direction because of the global information required. In the case of the convex network flow optimization, the equation defining the Newton direction is a node dimensional discrete Poisson equation (with constant input)– a linear system of equations characterized by a weighted graph Laplacian. A study of the solutions of Laplacian linear equations via sparsifiers is presented in [Spielman, 2012]. The most recent methods found in [Peng and Spielman, 2014] are based on sparse matrix factorizations. The development of distributed implementations of sparsification based solvers would lead to new Newton-type distributed optimization methods.

In Chapter 2, a consensus based primal-dual Newton method first developed in [Jadbabaie et al., 2009] is presented. Consensus-based schemes have been used extensively in recent literature as distributed mechanisms for aligning the values held by multiple agents; see [Jadbabaie et al., 2003], [Olfati-Saber and Murray, 2004], and [Olshevsky and Tsitsiklis, 2006]. Consensus schemes converge assymptotically, resulting in an approximate Newton method. Further study of convergence rates for inexact Newton methods in [Dembo et al., 1982] and [Kelley, 1995].

In the convex network flow optimization problem, implementing a distributed Newton method requires computation of the Newton direction within every dual iteration. Both the consensus based and sparsification based solvers are iterative methods requiring communication resources, making the overall distributed Newton-type methods extremely costly in terms of communication. Limiting the conensus scheme to a finite number of iterations is presented in [Zargham et al., 2014c]. There is a fundamental tradeoff between communication and computation which is explored in [Tsianos et al., 2012]. In [Zargham et al., 2014c], it is shown that when approximating the Newton method for convex network flow, two or fewer consensus iterations are necessary to compute a sufficiently accurate Newton direction. The finite truncation of the consensus based Newton's method is called Accelerated Dual descent and is analyzed extensively in Chapter 3.

1.3 Packet Routing as Network Flow Optimization

The apparent instantaneous availability of data in the modern age is made possible by a complex network of routers, cables, servers and other physical devices. This infrastructure is responsible for routing packets of data from its host location to a request location. The natural application to consider is internet service, but distributed systems such as vehicle coordination (e.g. [Fink et al., 2012]) and sensor fusion problems (e.g. [Al-Karaki and Kamal, 2004]) have underyling coordination tasks which require efficient communication often in the presence of noisy wireless channels. The main thrust of the packet routing literature is motivated by wireless networking: [Tassiulas and Ephremides, 1992], [Neely et al., 2005], and [Georgiadis et al., 2006]. In this section, the development of Backpressure routing is reviewed and a connection to the stochastic convex network flow optimizaion problem is established.

1.3.1 Queue Stabilization Literature

The queue stabilization problem models packet routing as continuous flows over communication links with with fixed capacity in a network which supports multiple data flows. The objective of the stabilization is to route all packets which enter the network to their destination. Each node maintains a queue of packets of each type which may accumulate if the packets routed into the node exceed those routed out. The queue stabilization problem takes its name from that guaranteeing eventual packet delivery is equivalent to guranteeing the stability of the packet queues at every node.

The Backpressure Algorithm introduced in [Tassiulas and Ephremides, 1992] addresses the queue stabilization problem using an analogy to pnuematic pressure in fluid systems. Flows are routed over the links based on the queue length differentials, allocating the full link capacity to the data flow with the greatest differential. Since the queue differential is a local variable, Backpressure can be implemented directly using local protocols at the nodes, requiring communication with immediate neighbors only. Backpressure is shown to be through-put optimal meaning that it is guaranteed to prevent any queue from becoming unbounded. The analysis provided assumes stochastic packet arrival rates and uses a Lyaponuv drift argument to guarantee queue stability.

The study of the Backpressure algorithm is extended in [Neely et al., 2005]. The concept of the capacity region is introduced in order to analyze the conditions under which it is possible to solve the queue stabilization problem when each node has limited power available for transmission. Emphasis is placed on the dynamic nature of routing control generated by the backpressure algorithm and stability analysis is presented using the Lyapunov drift technique. In [Eryilmaz and Srikant, 2005], it is shown that Backpressure not only stabilizes the queues but leads to fair resourse allocation with respect to the end users, represented by data types with unique destination nodes. Fairness of the Backpressure algorithm is independently addressed in [Neely et al., 2008].

1.3.2 Connection to Convex Network Flow Optimization

Network flow optimization and queue stabilization both solve a network resource allocation problem. Unlike the network flow optimization problem the packet routing problem explicitly allows the conservation constraint at each node to be violated at any point in time. Whenever the conservation constraint is violated, the untransmitted packets are stored in a queue to be transmitted in a later time slot. In order for a queue to remain stable, it is necessary that the conservation constraint hold in expectation, [Ribeiro, 2009a]. The packet routing problem can be stated as a stochastic capacity constrained network flow optimization problem where the conservation constraint is stated using the expected arrival rates at each node. In [Gatsis et al., 2010], it is shown that dual stochastic subgradient can be used to solve a packet routing problem using observations of the instantaneous conservation constraint violation.

When solving the packet routing problem via an optmization framework the convergence rate of the methods used determines the stability properties of the queues. Optimization based methods such as stochastic subgradient descent are iterative methods which approach the optimal point asymptotically. A suboptimal routing policy may not satisfy the conservation constraint allowing the queues to build up. When analyzing optimization based methods, one must demonstrate queue stability as a benchmark. In practice, optimization algorithms which converge faster result in significantly shorter queues at steady state. In chapters 4 and 5, fast distributed optimization methods are implemented to solve the packet routing problem preventing the queues from accumulating.

1.3.3 Optimization based Methods for Packet Routing

In [Ribeiro, 2009b], the Backpressure algorithm is rederived as dual stochastic subgradient descent applied to a feasibility problem closely related to the network flow problem. This work introduces Soft Backpressure (SBP), a generalization of of the Backpressure algorithm, which allows the user to select a convex objective function resulting in a convex network flow optimization problem. Since the original problem is a feasibility problem, the optimal solution with any objective is still a solution to the feasibility problem; a well choosen objective can help drive the system to feasible point much faster. Including a quadratic objective function results in resource sharing; rather than always allocate all of a links capacity to the flow with the greatest queue differential, the capacity is divided in proportion to the queue differentials. Provided that the objective function chosen is separable in the edges, dual stochastic gradient descent can be implemented using local protocols and is equivalent to Soft Backpressure.

In Section 1.2, it was established that subgradient descent has slow convergence. Having recast the packet routing problem as a convex network flow optimization more advanced methods can be implemented in order to achieve faster convergence with local protocols. The Accelerated Dual Descent (ADD) method is generalized for stochastic and capacity constrainted problems in [Zargham et al., 2012] and [Zargham et al., 2013b], respectively. Applying the ADD method to the convex network flow problem introduced [Ribeiro, 2009b] yields a new backpressure generalization called Accelerated Backpressure (ABP).

Accelerated Backpressure differs from Backpressure and Soft Backpressure because the routings are computed using a set of queue priorities which are Langrange dual variables rather than simply the queue lengths. These priorities take into account curvature information because they are computed using ADD, a distributed approximation to the Newton Method. In [Zargham et al., 2014b], ABP is shown to exceed the queue stabilization benchmark, emptying the queues infinitely often when implemented with a decaying stepsize. In Chapter 4, queue stability analysis of ABP is provided using a supermartingale convergence argument rather than Lyaponov drift analysis. Supermartingale convergence analysis is better suited to the problem of tracking both queue lengths and queue priorities (Lagrange dual variables), both of which are random sequences driven by the realization of the stochastic packet arrival process.

Ongoing research in the area of optimization based methods for packet routing problems includes the introduction of a discounted integrator to the SBP algorithm. Motivated by the control literature, the introduction of an integral controller reduces steady state error, in the context of queue stabilization this means driving down the steady queue lengths. This method shares similarities with the heavy ball method introduced in [Nesterov, 1983], but is novel in its application to packet routing problems.

Derivation of the discounted integrator method for queue stabilization as well as preliminary numerical and analytical results are presented in [Zargham et al., 2014a]. In Chapter 5, these derivations and results are presented with complete proofs. In order to provide stability analysis, a modified version of the discounted integral priority routing algorithm is modeled after the stochastic heavy ball method in [Flam, 2004]. This method allows the implementation of a decaying stepsize consistent with the convergence analysis of stochastic gradient descent. The numerical results demonstrate performance with a unit stepsize. Without the decaying stepsize, discounted integral priority routing can be interpreted as a variant of stochastic average gradient (see [Schmidt et al., 2013]); rather than giving equal weight to a history of stochastic gradients, the discounted integral method puts more weight on newer stochastic gradients.

Chapter 2

A Consensus based Newton Method

The basic approach to distributed optimization in networks is to use subgradient methods, which yield iterative algorithms that operate on the basis of local information. A major shortcoming of this approach, particularly relevant in today's large-scale networks, is the slow convergence rate of the resulting algorithms. In this chapter, an alternative approach based on using Newton-type methods is proposed for minimum cost network optimization problems. The proposed method can be implemented in a distributed manner and has faster convergence properties.

The challenges in applying Newton-type methods in this context are twofold. First, the superlinear convergence rate of this type of method is achieved by using a backtracking stepsize rule, which requires global information in the form of the norm of a residual vector. This problem is addressed by using a consensus-based local averaging scheme for estimating the norm of the residual. Second, the computation of the dual Newton step involves a matrix inversion, which requires global information. The main contribution is to develop a distributed iterative scheme for the computation of the dual Newton step. The key idea is to recognize that the dual Newton step is defined through a discrete Poisson equation (with constant input), which involves the Laplacian of the graph, and therefore can be solved using a consensus-based scheme in a distributed manner. We show that the convergence rate of this scheme is governed by the spectral properties of the underlying graph. For fast-mixing graphs (i.e., those with large spectral gaps), the dual Newton step can be computed efficiently using only local information.

Since our method uses consensus-based schemes to compute the stepsize and the Newton direction in each iteration, exact computation is not feasible. Another contribution of this chapter is to consider truncated versions of these consensus-schemes at each iteration and present convergence rate analysis of the constrained Newton method when the stepsize and the direction are estimated with some error. We show that when these errors are sufficiently small, the value of the residual function converges superlinearly to a neighborhood of the origin, whose size is explicitly quantified as a function of the errors and the parameters of the objective function and the constraints of the minimum cost network flow optimization problem.

2.1 Convex Network Flow Optimization Problem

Consider a network represented by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. Each edge e in the network has a convex cost function $\phi_e(x^e)$, which captures the cost due to congestion effects as a function of the flow x^e on this edge. The total cost of a flow vector $x = [x^e]_{e\in\mathcal{E}}$ is given by the sum of the edge costs, i.e., $\sum_{e\in\mathcal{E}} \phi_e(x^e)$. Given an external supply b_i for each node $i \in \mathcal{N}$, the convex network flow optimization problem is to find a minimum cost flow allocation vector that satisfies the flow conservation constraint at each node.[1] This problem can be formulated as a convex optimization problem with linear equality constraints. The application of dual decomposition together with a dual subgradient algorithm then yields a distributed iterative solution method. Instead, we propose a distributed primal-dual Newton-type method that achieves a superlinear convergence rate.

2.1.1 Preliminaries

A vector is viewed as a column vector, unless clearly stated otherwise. We denote by x^i the i-th component of a vector x. When $x^i \geq 0$ for all components i of a vector x, we write $x \geq 0$. For a matrix A, we write A_{ij} or $[A]_{ij}$ to denote the matrix entry in the i-th row and j-th column. We write

[1]We focus on feasible problems, i.e., we assume that the total in-flow to the network is equal to the total out-flow, $\sum_{i\in\mathcal{N}} b_i = 0$.

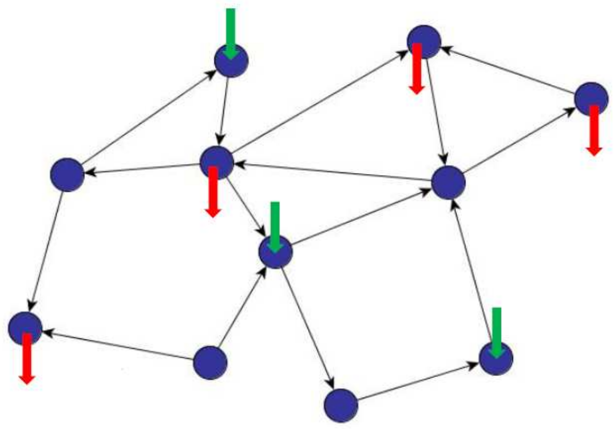


Figure 2.1: The network flow optimization problem consists of a directed network with an external supply vector encoding the sink and source nodes. A feasible solution to this problem is an edge dimension vector of flow rates which satisfies a conservation constraint at each node.

x' to denote the transpose of a vector x. The scalar product of two vectors $x, y \in \mathbb{R}^m$ is denoted by $x'y$. We use $\|x\|$ to denote the standard Euclidean norm, $\|x\| = \sqrt{x'x}$. For a vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$, the gradient matrix of f at $x \in \mathbb{R}^n$ is denoted by $\nabla f(x)$.

A vector $a \in \mathbb{R}^m$ is said to be a *stochastic vector* when its components a_i, $i = 1, \dots, m$, are nonnegative and their sum is equal to 1, i.e., $\sum_{i=1}^m a_i = 1$. A square $m \times m$ matrix A is said to be a *stochastic matrix* when each row of A is a stochastic vector. A stochastic matrix is called irreducible and aperiodic if all eigenvalues (except the eigenvalue at 1) are subunit.

One can associate a discrete-time Markov chain with a stochastic matrix and a graph $\mathcal{G}$ as follows: The state of the chain at time $k \in \{1, 2, \cdots\}$, denoted by $X(k)$, is a node in $\mathcal{N}$ (the node set of the graph) and the weight associated to each edge in the graph is the probability with which X makes a transition between two adjacent nodes. In other words, the transition from state i to state j happens with probability p_{ij}, the weight of edge (i, j). If $\pi(k)$ with elements defined as $\pi_i(k) = \mathbb{P}(X(k) = i)$ is the probability distribution of the state at time k, the state distribution satisfies the recursion $\pi(k+1)^T = \pi(k)^T P$. If the chain is irreducible and aperiodic then for all initial distributions, π converges to the unique stationary distribution π^* [Berman and Plemmons, 1979].

2.1.2 Problem Formulation

We consider a network represented by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with node set $\mathcal{N} = \{1, \ldots, N\}$, and edge set $\mathcal{E} = \{1, \ldots, E\}$. We denote the flow vector by $x = [x^e]_{e\in\mathcal{E}}$, where x^e denotes the flow on edge e. The flow conservation conditions at the nodes can be compactly expressed as$Ax = b$, where A is the $N \times E$ *node-edge incidence matrix* of the graph, i.e.,

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

and the vector b denotes the external sources, i.e., $b_i > 0$ (or $b_i < 0$) indicates b_i units of external flow enters (or exits) node i. We associate a cost function $\phi_e : \mathbb{R} \to \mathbb{R}$ with each edge e, i.e., $\phi_e(x^e)$ denotes the cost on edge e as a function of the edge flow x^e. We assume that the cost functions ϕ_e are strictly convex and twice continuously differentiable. The minimum cost network optimization problem can be written as

$$\text{minimize} \sum_{e=1}^{E} \phi_e(x^e) \quad \text{subject to: } Ax = b \tag{2.1}$$

In this paper, our goal is to investigate *iterative distributed methods* for solving problem (2.1). In particular, we focus on two methods: first relies on solving the dual of problem (2.1) using a subgradient method; second uses a constrained Newton method, where, at each iteration, the Newton direction is computed iteratively using an averaging method.

2.1.3 Dual Subgradient Method

We first consider solving problem (2.1) using a dual subgradient method. To define the dual problem, we form the Lagrangian function of problem (2.1) $\mathcal{L} : \mathbb{R}^E \times \mathbb{R}^N \to \mathbb{R}$ given by $\mathcal{L}(x,\lambda) = \sum_{e=1}^{E} \phi_e(x^e) - \lambda'(Ax - b)$. The dual function $q(\lambda)$ becomes

$$\begin{aligned} q(\lambda) &= \inf_{x\in\mathbb{R}^E} \mathcal{L}(x,\lambda) = \inf_{x\in\mathbb{R}^E} \left(\sum_{e=1}^{E} \phi_e(x^e) - \lambda' A x \right) + \lambda' b \\ &= \sum_{e=1}^{E} \inf_{x^e\in\mathbb{R}} \Big(\phi_e(x^e) - (\lambda' A)^e x^e \Big) + \lambda' b. \end{aligned}$$

Hence, in view of the fact that the objective function and the constraints of problem (2.1) are separable in the decision variables x^e, the evaluation of the dual function decomposes into one-dimensional optimization problems. We assume that each of these optimization problems has an optimal solution, which is unique by the strict convexity of the functions ϕ_e and is denoted by $x^e(\lambda)$. Using the first order optimality conditions, it can be seen that for each e, $x^e(\lambda)$ is given by

$$x^e(\lambda) = (\phi_e')^{-1}(\lambda^i - \lambda^j), \tag{2.2}$$

where $i, j \in \mathcal{N}$ denote the end nodes of edge e. Thus, for each edge e, the evaluation of $x^e(\lambda)$ can be done based on local information about the edge cost function ϕ^e and the dual variables of the incident nodes i and j. We can write the dual problem as $\max_{\lambda\in\mathbb{R}^N}\ q(\lambda)$. The dual problem can be solved by using a subgradient method: given an initial vector λ_0, the iterates are generated by $\lambda_{k+1} = \lambda_k - \alpha_k g_k \qquad$ for all $k \geq 0$, where g_k is a subgradient of the dual function $q(\lambda)$ at $\lambda = \lambda_k$ given by $g_k = Ax(\lambda_k) - b$, and $x(\lambda_k) = \operatorname{argmin}_{x\in\mathbb{R}^E} \mathcal{L}(x, \lambda_k)$, i.e., for all $e \in \mathcal{E}$, $x^e(\lambda_k)$ is given by Eq. (2.2) with $\lambda = \lambda_k$.

This method naturally lends itself to a distributed implementation: each node i updates its dual variable λ^i using local (subgradient) information g^i obtained from edges e incident to that node, which in turn updates its primal variables $x^e(\lambda)$ using dual variables of the incident nodes. Despite its simplicity and distributed nature, however, it is well-known that the dual subgradient method suffers from slow rate of convergence (see [Nedić and

Ozdaglar, 2008] and [Nedić and Ozdaglar, 2009a] for rate analysis and construction of primal solutions for dual subgradient methods), which motivates us to consider a Newton method for solving problem (2.1).

2.1.4 Equality-Constrained Newton Method

Consider a solution to the problem (2.1) using an (infeasible start) equality-constrained Newton method (see [Boyd and Vandenberghe, 2004], Chapter 10). We let $f(x) = \sum_{e=1}^{E} \phi_e(x^e)$ for notational simplicity. Given an initial primal vector x_0, the iterates are generated by $x_{k+1} = x_k + \alpha_k v_k$ where v_k is the Newton step given as the solution to the following system of linear equations:[2]

$$\begin{pmatrix} \nabla^2 f(x_k) & A' \\ A & 0 \end{pmatrix} \begin{pmatrix} v_k \\ w_k \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) \\ Ax_k - b \end{pmatrix}.$$

We let $H_k = \nabla^2 f(x_k)$ and $h_k = Ax_k - b$ for notational convenience. Solving for v_k and w_k in the preceding yields $v_k = -H_k^{-1}(\nabla f(x_k) + A'w_k)$, and

$$(AH_k^{-1}A')w_k = h_k - AH_k^{-1}\nabla f(x_k). \tag{2.3}$$

Since the matrix H_k^{-1} is a diagonal matrix with entries $[H_k^{-1}]_{ee} = (\frac{\partial^2 \phi_e}{(\partial x^e)^2})^{-1}$, given the vector w_k, the Newton step v_k can be computed using local information. However, the computation of the vector w_k at a given primal vector x_k cannot be implemented in a decentralized manner in view of the fact that solving equation (2.3) $(AH_k^{-1}A')^{-1}$ requires global information. The following section provides an iterative scheme to compute the vector w_k using local information.

[2]This is essentially a primal-dual method with the vectors v_k and w_k acting as primal and dual steps; see Section 2.2.

2.1.5 Distributed Computation of the Newton Direction

Consider the vector w_k defined in Eq. (2.3). The key step in developing a decentralized iterative scheme for the computation of the vector w_k is to recognize that the matrix $AH_k^{-1}A'$ is the weighted Laplacian of the underlying graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, denoted by L_k. Hence, L_k can be written as $L_k = AH_k^{-1}A' = D_k - B_k$. Here B_k is an $N \times N$ matrix with entries

$$(B_k)_{ij} = \begin{cases} \left(\frac{\partial^2 \phi_e}{(\partial x^e)^2}\right)^{-1} & \text{if } e = (i,j) \in \mathcal{E}, \\ 0 & \text{otherwise,} \end{cases} \tag{2.4}$$

and D_k is an $N \times N$ diagonal matrix with entries $(D_k)_{ii} = \sum_{j \in \mathcal{N}_i} (B_k)_{ij}$, where $\mathcal{N}_i$ denotes the set of neighbors of node i, i.e., $\mathcal{N}_i = \{j \in \mathcal{N} \mid (i,j) \in \mathcal{E}\}$. Letting $s_k = h_k - AH_k^{-1}\nabla f(x_k)$ for notational convenience, Eq. (2.3) can be then rewritten as

$(I - (D_k + I)^{-1}(B_k + I))w_k = (D_k + I)^{-1}s_k$. This motivates the following iterative scheme (known as splitting) to solve for w_k. For any $t \geq 0$, the iterates are generated by

$w(t+1) = (D+I)^{-1}(B+I)w(t) + (D+I)^{-1}s$, (where we suppressed the indices k for notational convenience). Note that the matrix $P := (D+I)^{-1}(B+I)$ is row stochastic, and when the graph of the network is connected, it is irreducible and aperiodic (i.e., a primitive matrix) [Horn and Johnson, 1985]. Furthermore, P is diagonally similar to a symmetric matrix, therefore all of its eigenvalues are real and by Perron-Frobenius theorem [Berman and Plemmons, 1979] they satisfy

$$1 = \lambda_1(P) > \lambda_2(P) \geq \cdots \geq \lambda_n(P) > -1. \tag{2.5}$$

Aside from one eigenvalue at 1 (corresponding to the all-one eigenvector $\mathbf{1}$), all other eigenvalues are subunit [Horn and Johnson, 1985]. As a result, the projection of the dynamics to the orthogonal complement of the span of $\mathbf{1}$ is stable. Let V be the $n \times (n-1)$ dimensional matrix whose orthonormal $n-1$ columns span $\mathbf{1}^{\perp}$, the orthogonal subspace to $\mathbf{1}$. Denote

$w(t) = V\bar{w}(t), s = V\bar{s}$, where $V'V = I_{n-1}$, and $VV' = I_n - \frac{\mathbf{1}\mathbf{1}^T}{n}$. The projected dynamics can now be written as

$$\bar{w}(t+1) = V'PV\bar{w}(t) + V'(D+I)^{-1}V\bar{s}. \tag{2.6}$$

Equation (2.6) depicts the dynamics of a stable linear system with a constant (step) input, which from basic linear systems theory is known to converge to the solution of equation (2.3). The exponential convergence rate is determined by the convergence rate of the random walk $w(t+1) = Pw(t)$ to its stationary distribution. More precisely, the speed of convergence of $w(t)$ to the stationary distribution, depends on the eigenstructure of the probability transition (weight) matrix P.

2.1.6 Convergence Rates

It is well known that the rate of convergence to the stationary distribution of a Markov chain is governed by the second largest eigenvalue modulus of matrix P defined as

$$\mu(P) = \max_{i=2,\cdots,n}\{|\lambda_i(P)|\} = \max\{\lambda_2(P), |\lambda_n(P)|\}. \tag{2.7}$$

To make this statement more precise, let i be the initial state and define the *total variation distance* between the distribution at time k and the stationary distributionπ^* as

$$\Delta_i(k) = \frac{1}{2}\sum_{j\in\mathcal{V}} |P_{ij}^k - \pi_j^*|. \tag{2.8}$$

The rate of convergence to the stationary distribution is measured using the following quantity known as the *mixing time*:

$$T_{mix} = \max_i \min\{k : \Delta_i(k') < e^{-1} \text{ for all } k' \geq k\}. \tag{2.9}$$

The following theorem indicates the relationship between the mixing time of a Markov chain and the second largest eigenvalue modulus of its probability transition matrix [Aldous, 1982; Sinclair, 1992].

Theorem 1. *The mixing time of a reversible Markov chain with transition probability matrix W and second largest eigenvalue modulus μ satisfies*

$$\frac{\mu}{2(1-\mu)}(1-\ln 2) \leq T_{mix} \leq \frac{1+\log n}{1-\mu}.$$

Therefore, the speed of convergence of the Markov chain to its stationary distribution is determined by the value of $1-\mu$ known as the *spectral gap*; the larger the spectral gap, the faster the convergence. This suggests that for each iteration k, the dual Newton step w_k can be computed faster on graphs with large spectral gap.

The above convergence results indicate that the dual Newton step can be computed in a decentralized fashion with a diffusion scheme, or more precisely, with a discrete Poisson equation. Since our distributed solution involves an iterative scheme, exact computation is not feasible. In what follows, we show that the Newton method has desirable convergence properties even when the Newton direction is computed with some error provided that the errors are sufficiently small.

2.2 Inexact Newton Method

In this section, we consider the following convex optimization problem with equality constraints:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & Ax = b, \end{aligned} \tag{2.10}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a twice continuously differentiable convex function, and A is an $m \times n$ matrix. The minimum cost network optimization problem (2.1) is a special case of this problem with $f(x) = \sum_{e=1}^{E} \phi_e(x^e)$ and A is the $N \times E$ node-edge incidence matrix. We denote the optimal value of this problem by f^*. Throughout this section, we assume that the value f^* is finite and problem (2.10) has an optimal solution, which we denote by x^*.

We consider an *inexact (infeasible start) Newton method* for solving problem (2.10) (see [Boyd and Vandenberghe, 2004]). In particular, we let $y = (x, \nu) \in \mathbb{R}^n \times \mathbb{R}^m$, where x is the primal variable and ν is the dual variable, and study a primal-dual method which updates the vector y at iteration k as follows:

$$y_{k+1} = y_k + \alpha_k d_k, \tag{2.11}$$

where α_k is a positive stepsize, and the vector d_k is an *approximate constrained Newton direction* given by

$$Dr(y_k)d_k = -r(y_k) + \epsilon_k. \tag{2.12}$$

Here, the residual function $r : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^m$ is defined as

$$\begin{aligned} r(x,\nu) &= (r_{dual}(x,\nu), r_{pri}(x,\nu)), && (2.13)\\ r_{dual}(x,\nu) &= \nabla f(x) + A'\nu && (2.14)\\ r_{pri}(x,\nu)) &= Ax - b. && (2.15) \end{aligned}$$

Moreover, $Dr(y) \in \mathbb{R}^{(n+m)\times(n+m)}$ is the gradient matrix of r evaluated at y, and the vector ϵ_k is an error vector at iteration k. We assume that the error sequence $\{\epsilon_k\}$ is uniformly bounded from above, i.e., there exists a scalar $\epsilon \geq 0$ such that

$$\|\epsilon_k\| \leq \epsilon \text{ for all } k \geq 0. \tag{2.16}$$

We adopt the following standard assumption:

Assumption 1. *Let $r : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^m$ be the residual function defined in Eqs. (2.13)-(2.15). Then, we have:*

(a) (Lipschitz Condition) There exists some constant $L > 0$ such that

$$\|Dr(y) - Dr(\bar{y})\| \leq L\|y - \bar{y}\| \, \forall y, \bar{y} \in \mathbb{R}^n \times \mathbb{R}^m.$$

(b) There exists some constant $M > 0$ such that

$$\|Dr(y)^{-1}\| \leq M \qquad \forall y \in \mathbb{R}^n \times \mathbb{R}^m.$$

2.2.1 Basic Relation

We use the norm of the residual vector $\|r(y)\|$ to measure the progress of the algorithm. In the next proposition, we present a relation between the iterates $\|r(y_k)\|$, which holds for any stepsize rule. The proposition follows from a multi-dimensional extension of the descent lemma (see [Bertsekas et al., 2003]).

Proposition 1. *Let Assumption 1 hold. Let $\{y_k\}$ be a sequence generated by the method (2.11). For any stepsize rule α_k, we have*

$$\|r(y_{k+1})\| \leq (1 - \alpha_k)\|r(y_k)\| + M^2 L\alpha_k^2\|r(y_k)\|^2 + \alpha_k\|\epsilon_k\| + M^2 L\alpha_k^2\|\epsilon_k\|^2.$$

Proof : We consider two vectors $w \in \mathbb{R}^n \times \mathbb{R}^m$ and $z \in \mathbb{R}^n \times \mathbb{R}^m$. We let ξ be a scalar parameter and define the function $g(\xi) = r(w + \xi z)$. From the chain rule, it follows that $\nabla g(\xi) = Dr(w + \xi z)z$. Applying this equality and the fundamental theorem of calculus,

$$\begin{aligned} r(w+z) - r(w) &= g(1) - g(0) && (2.17)\\ &= \int_0^1 \nabla g(\xi) d\xi && (2.18)\\ &= \int_0^1 Dr(w + \xi z) z d\xi. && (2.19) \end{aligned}$$

Adding and subtracting $\int_0^1 Dr(w)zd\xi$ and taking an absolute value over the two leading terms yields the inequality

$$r(w+z) - r(w) \leq \left| \int_0^1 (Dr(w+\xi z) - Dr(w))zd\xi \right| + \int_0^1 Dr(w)zd\xi. \tag{2.20}$$

Using Jensen's inequality on (2.20), we have

$$r(w+z) - r(w) \leq \int_0^1 \|Dr(w+\xi z) - Dr(w)\| \, \|z\| d\xi + Dr(w)z. \tag{2.21}$$

Applying Cauchy-Schwartz inequality allows us to factor our $||z||$ and apply the Lipschitz continuity of the residual function gradient [cf. Assumption 1(a)],

$$\begin{aligned} r(w+z) - r(w) &\leq \|z\| \int_0^1 L\xi \|z\| d\xi + Dr(w)z && (2.22)\\ &= \frac{L}{2}\|z\|^2 + Dr(w)z. && (2.23) \end{aligned}$$

We apply equation (2.23) with $w = y_k$ and $z = \alpha_k d_k$ to obtain

$$r(y_k + \alpha_k d_k) - r(y_k) \leq \alpha_k Dr(y_k) d_k + \frac{L}{2}\alpha_k^2 \|d_k\|^2. \tag{2.24}$$

By Eq. (2.12), we have $Dr(y_k)d_k = -r(y_k) + \epsilon_k$. Substituting this into (2.24) yields

$$r(y_k + \alpha_k d_k) \leq (1 - \alpha_k)r(y_k) + \alpha_k \epsilon_k + \frac{L}{2}\alpha_k^2 \|d_k\|^2. \tag{2.25}$$

Moreover, using Applying (2.12), isolating d_k and applying the $||\cdot||^2$ operator yields

$$\|d_k\|^2 = \|Dr(y_k)^{-1}(-r(y_k) + \epsilon_k)\|^2. \tag{2.26}$$

Applying the Cauchy-Schwartz inequality and Assumption 1(b), we have

$$\begin{aligned} \|d_k\|^2 &\leq \|Dr(y_k)^{-1}\|^2 \, \| - r(y_k) + \epsilon_k\|^2 && (2.27)\\ &\leq M^2\Big(2\|r(y_k)\|^2 + 2\|\epsilon_k\|^2\Big). && (2.28) \end{aligned}$$

Combining the above relations, we obtain

$$\|r(y_{k+1})\| \leq (1 - \alpha_k)\|r(y_k)\| + M^2 L\alpha_k^2 \|r(y_k)\|^2 + \alpha_k\|\epsilon_k\| + M^2 L \alpha_k^2 \|\epsilon_k\|^2, \tag{2.29}$$

establishing the desired relation. ■

2.2.2 Inexact Backtracking Stepsize Rule

We use a backtracking stepsize rule in our method to achieve the superlinear local convergence properties of the Newton method. However, this requires computation of the norm of the residual function $\|r(y)\|$. In view of the distributed nature of the residual vector $r(y)$ [cf. Eq. (2.13)], this norm can be computed using a distributed consensus-based scheme. Since this scheme is iterative, in practice the residual norm can only be estimated with some error.

Let $\{y_k\}$ be a sequence generated by the inexact Newton method (2.11). At each iteration k, we assume that we can compute the norm $\|r(y_k)\|$ with some error, i.e., we can compute a scalar $n_k \geq 0$ that satisfies

$$\left| n_k - \|r(y_k)\| \right| \leq \gamma/2, \tag{2.30}$$

for some constant $\gamma \geq 0$. Hence, n_k is an approximate version of $\|r(y_k)\|$, which can be computed for example using distributed iterative methods. For fixed scalars $\sigma \in (0, 1/2)$ and $\beta \in (0, 1)$, we set the stepsize α_k equal to $\alpha_k = \beta^{m_k}$, where m_k is the smallest nonnegative integer that satisfies

$$n_{k+1} \leq (1 - \sigma\beta^m)n_k + B + \gamma. \tag{2.31}$$

Here, γ is the maximum error in the residual function norm [cf. Eq. (2.30)], and B is a constant given by

$$B = \epsilon + M^2 L \epsilon^2, \tag{2.32}$$

where ϵ is the upper bound on the error sequence in the constrained Newton direction [cf. Eq. (2.16)] and M and L are the constants in Assumption 1.

2.2.3 Convergence Rate for Damped Newton Phase

We first show a strict decrease in the norm of the residual function if the errors ϵ and γ are sufficiently small, as quantified in the following assumption.

Assumption 2. *The errors B and e [cf. Eqs. (2.32) and (2.30)] satisfy*

$$B + 2\gamma \leq \frac{\beta}{16M^2L},$$

where β is the constant used in the inexact backtracking stepsize rule, and M and L are *the constants defined in Assumption 1.*

Under this assumption, the next proposition establishes a strict decrease in the norm of the residual function as long as $\|r(y)\| > \frac{1}{2M^2L}$.

Proposition 2. *Let Assumptions 1 and 2 hold. Let $\{y_k\}$ be a sequence generated by the method (2.11) when the stepsize sequence $\{\alpha_k\}$ is selected using the inexact backtracking stepsize rule [cf. Eq. (2.31)]. Assume that $\|r(y_k)\| > \frac{1}{2M^2L}$. Then, we have*

$$\|r(y_{k+1})\| \leq \|r(y_k)\| - \frac{\beta}{16M^2L}.$$

Proof : For any $k \geq 0$, we define

$$\bar{\alpha}_k = \frac{1}{2M^2L(n_k + \gamma/2)}. \tag{2.33}$$

In view of the condition on n_k [cf. Eq. (2.30)], we have

$$\frac{1}{2M^2L(\|r(y_k)\| + \gamma)} \leq \bar{\alpha}_k \leq \frac{1}{2M^2L\|r(y_k)\|} < 1, \tag{2.34}$$

where the last inequality follows by the assumption $\|r(y_k)\| > \frac{1}{2M^2L}$. Substituting $\alpha_k = \bar{\alpha}_k$ in the basic relation in Proposition 1, we obtain:

$$\|r(y_{k+1})\| \leq \|r(y_k)\| + \bar{\alpha}_k\|\epsilon_k\| + M^2L\bar{\alpha}_k^2\|\epsilon_k\|^2 - \bar{\alpha}_k\|r(y_k)\|\Big(1 - M^2L\bar{\alpha}_k\|r(y_k)\|\Big). \tag{2.35}$$

Applying the second relation in (2.34),

$$\begin{aligned} \|r(y_{k+1})\| \quad \leq \quad & \|r(y_k)\| + \bar{\alpha}_k\|\epsilon_k\| + M^2L\bar{\alpha}_k^2\|\epsilon_k\|^2 \\ & -\bar{\alpha}_k\|r(y_k)\|\Big(1 - M^2L\frac{\|r(y_k)\|}{2M^2L\|r(y_k)\|}\Big) \end{aligned} \tag{2.36}$$

. Simplifying by applying the defintion of $\bar{\alpha}$ from (2.33) yields

$$\|r(y_{k+1})\| \leq \bar{\alpha}_k \|\epsilon_k\| + M^2 L \bar{\alpha}_k^2 \|\epsilon_k\|^2 + \Big(1 - \frac{\bar{\alpha}_k}{2}\Big)\|r(y_k)\|. \tag{2.37}$$

From (2.34), $\bar{\alpha} \leq 1$ so applying the definition of B in (2.32) yields

$$\|r(y_{k+1})\| \leq B + \Big(1 - \frac{\bar{\alpha}_k}{2}\Big)\|r(y_k)\|. \tag{2.38}$$

The constant σ used in the definition of the inexact backtracking line search satisfies $\sigma \in (0, 1/2)$, therefore, it follows from the preceding relation that

$$\|r(y_{k+1})\| \leq (1 - \sigma\bar{\alpha}_k)\|r(y_k)\| + B. \tag{2.39}$$

Using condition (2.30), this implies

$$n_{k+1} \leq (1 - \sigma\bar{\alpha}_k) n_k + B + \gamma, \tag{2.40}$$

showing that the steplength α_k selected by the inexact backtracking line search satisfies $\alpha_k \geq \beta\bar{\alpha}_k$. From condition (2.31), we have

$$n_{k+1} \leq (1 - \sigma\alpha_k) n_k + B + \gamma, \tag{2.41}$$

which implies

$$\|r(y_{k+1})\| \leq (1 - \sigma\beta\bar{\alpha}_k)\|r(y_k)\| + B + 2\gamma. \tag{2.42}$$

Combined with Eq. (2.34), this yields

$$\|r(y_{k+1})\| \leq \Big(1 - \frac{\sigma\beta}{2M^2 L(\|r(y_k)\| + \gamma)}\Big)\|r(y_k)\| + B + 2\gamma. \tag{2.43}$$

By Assumption 2 and the fact that $\gamma \geq 0$, we have $\gamma \leq B + 2\gamma \leq \frac{\beta}{16M^2L}$, which in view of the assumption $\|r(y_k)\| > \frac{1}{2M^2L}$ implies that $\gamma \leq \|r(y_k)\|$. Substituting this into (2.43) and using the fact $\sigma \in (0, 1/2)$, we obtain

$$\|r(y_{k+1})\| \leq \|r(y_k)\| - \frac{\beta}{8M^2L} + B + 2\gamma. \tag{2.44}$$

Combined with Assumption 2, this yields the desired result. ■

The preceding proposition shows that, under the assumptions on the size of the errors, at each iteration, we obtain a minimum decrease (in the norm of the residual function) of $\frac{\beta}{16M^2L}$, as long as $\|r(y_k)\| > 1/2M^2L$. This establishes that we at most

$$\frac{16\|r(y_0)\|M^2L}{\beta}, \tag{2.45}$$

iterations are needed to reach the local convergence phase, defined as $\|r(y_k)\| \leq 1/2M^2L$.

2.2.4 Convergence Rate for Local Convergence Phase

In this section, we show that when $\|r(y_k)\| \leq 1/2M^2L$, the inexact backtracking stepsize rule selects a full step $\alpha_k = 1$ and the norm of the residual function $\|r(y_k)\|$ converges quadratically within an error neighborhood, which is a function of the parameters of the problem (as given in Assumption 1) and the error level in the constrained Newton direction.

Proposition 3. *Let Assumption 1 hold. Let $\{y_k\}$ be a sequence generated by the method (2.11) when the stepsize sequence $\{\alpha_k\}$ is selected using the inexact backtracking stepsize rule [cf. Eq. (2.31)]. Assume that there exists some k such that*

$$\|r(y_k)\| \leq \frac{1}{2M^2L}.$$

Then, the inexact backtracking stepsize rule selects $\alpha_k = 1$. We further assume that for some $\delta \in (0, 1/2)$,

$$B + M^2 L B^2 \leq \frac{\delta}{4M^2 L},$$

where B is the constant defined in Eq. (2.32). Then, we have

$$\|r(y_{k+m})\| \leq \frac{1}{2^{2^m} M^2 L} + B + \frac{\delta}{M^2 L} \frac{(2^{2^m - 1} - 1)}{2^{2^m}} \tag{2.46}$$

for all $m > 0$. As a particular consequence, we obtain

$$\limsup_{m \to \infty} \|r(y_m)\| \leq B + \frac{\delta}{2M^2 L}.$$

Proof : We first show that if $\|r(y_k)\| \leq \frac{1}{2M^2 L}$ for some $k > 0$, then the inexact backtracking stepsize rule selects $\alpha_k = 1$. Replacing $\alpha_k = 1$ in the basic relation of Proposition 1 and using the definition of the constant B defined in (2.32), we obtain

$$\begin{aligned} \|r(y_{k+1})\| &\leq M^2 L \|r(y_k)\|^2 + B & (2.47)\\ &\leq \frac{1}{2}\|r(y_k)\| + B. & (2.48) \end{aligned}$$

Recalling that $\sigma \in (0, 1/2)$ we can rewrite (2.48) as

$$\|r(y_{k+1})\| \leq (1 - \sigma)\|r(y_k)\| + B. \tag{2.49}$$

Using the condition on n_k [cf. Eq. (2.30)], this yields

$$n_{k+1} \leq (1 - \sigma) n_k + B + \gamma, \tag{2.50}$$

guaranteeing that the steplength $\alpha_k = 1$ satisfies condition (2.31) in the inexact backtracking stepsize rule.

We show Eq. (2.46) using induction on the iteration m. Using $\alpha_k = 1$ in the basic relation of Proposition 1, we obtain

$$\|r(y_{k+1})\| \leq \frac{1}{2}\|r(y_k)\| + B \leq \frac{1}{4M^2L} + B, \tag{2.51}$$

where the second inequality follows from the assumption $\|r(y_k)\| \leq \frac{1}{2M^2L}$. This establishes relation (2.46) for $m = 1$.

Given that (2.46) holds for some $m > 0$, and we show that that it also holds for $m + 1$. Eq. (2.46) implies that

$$\|r(y_{k+m})\| \leq \frac{1}{4M^2L} + B + \frac{\delta}{4M^2L}. \tag{2.52}$$

Using the assumption $B + M^2LB^2 \leq \frac{\delta}{4M^2L}$,

$$\|r(y_{k+m})\| \leq \frac{1+2\delta}{4M^2L} < \frac{1}{2M^2L}, \tag{2.53}$$

where the strict inequality follows from $\delta \in (0, 1/2)$. Hence, the inexact backtracking stepsize rule selects $\alpha_{k+m} = 1$. Using $\alpha_{k+m} = 1$ in the basic relation, we obtain

$$M^2L\|r(y_{k+m+1})\| \leq \Big(M^2L\|r(y_{k+m})\|\Big)^2 + B. \tag{2.54}$$

Applying Eq. (2.46) and simplifying yields

$$M^2L\|r(y_{k+m+1})\| \leq \left(\frac{1}{2^{2^m}} + M^2LB + \frac{\delta(2^{2^m-1}-1)}{2^{2^m}}\right)^2 + M^2LB \tag{2.55}$$

Writing out the square, we have

$$M^2L\|r(y_{k+m+1})\| \;\leq\; \left(\frac{1}{2^{2^m}}\right)^2 + 2\left(\frac{1}{2^{2^m}}\right)\left(M^2LB + \frac{\delta(2^{2^m-1}-1)}{2^{2^m}}\right) + \left(M^2LB + \frac{\delta(2^{2^m-1}-1)}{2^{2^m}}\right)^2 + M^2LB \tag{2.56}$$

and simplifying terms yields

$$M^2L\|r(y_{k+m+1})\| \;\leq\; \frac{1}{2^{2^{m+1}}} + \frac{M^2LB}{2^{2^m-1}} + \delta\,\frac{2^{2^m-1}-1}{2^{2^{m+1}-1}} + M^2L\left(B + \frac{\delta}{M^2L}\,\frac{(2^{2^m-1}-1)}{2^{2^m}}\right)^2 + M^2LB. \tag{2.57}$$

Using algebraic manipulations and the assumption $B + M^2LB^2 \leq \frac{\delta}{4M^2L}$, this yields

$$\|r(y_{k+m+1})\| \leq \frac{1}{2^{2^{m+1}}M^2L} + B + \frac{\delta}{M^2L}\,\frac{(2^{2^{m+1}-1}-1)}{2^{2^{m+1}}}, \tag{2.58}$$

completing the induction and therefore the proof of relation (2.46). Taking the limit superior in Eq. (2.46) establishes the final result. ■

2.3 Numerical Results

Our simulation results demonstrate that the decentralized Newton significantly outperforms the dual subgradient method algorithm in terms of runtime. Simulations were conducted as follows: Network flow problems with conservation constraints and the cost function $\Phi(x) = \sum_{e=1}^{E} \phi_e(x^e)$ where $\phi_e(x^e) = 1 - \sqrt{1-(x^e)^2}$ [3] were generated on Erdös-Rényi random graphs with $n = 10$, 20, 80, and 160 nodes and an expected node degree, $np = 5$.

[3]the cost function is motivated by the Kuramoto model of coupled nonlinear oscillators [Jadbabaie et al., 2004].

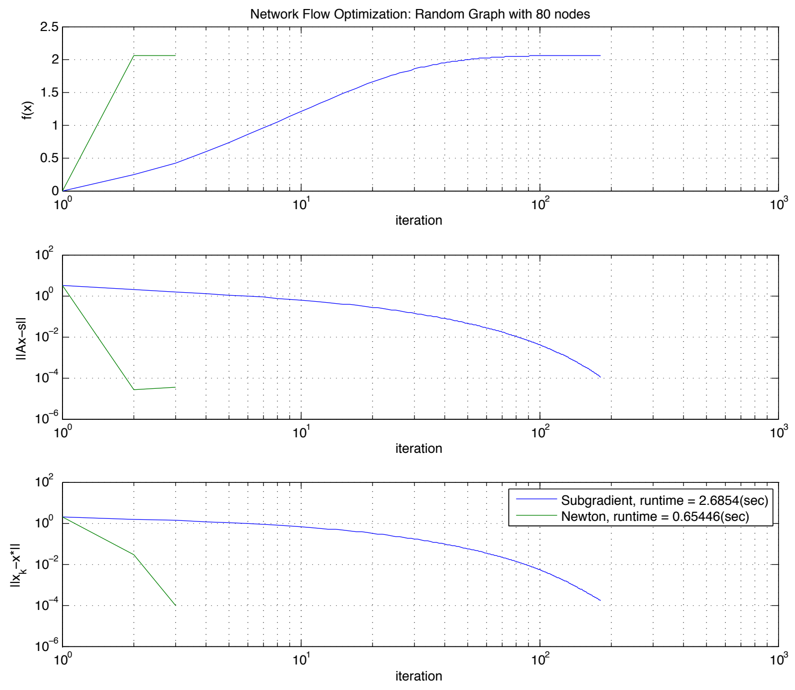


Figure 2.2: Sample convergence trajectory for a random graph with 80 nodes and mean node degree 5.

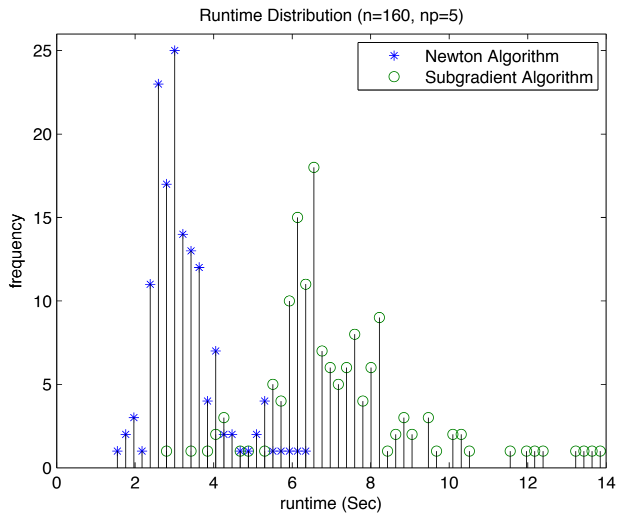
We limit the simulations to optimization problems which are well behaved in the sense that the Hessian matrix remains well conditioned, defined as $\frac{\lambda_{\max}}{\lambda_{\min}} \leq 200$. For this subclass of problem, the runtime of the Newton's method algorithm is significantly less than the subgradient method for all trials. Note that the stopping criterion is also tested in a distributed manner for both algorithms.

In any particular experiment the Newton's method algorithm discovers the feasible direction in one iteration and proceeds to find the optimal solution in two to four additional iterations. One such experiment is shown in Figure 2.2. A sample runtime distribution for 150 trials is presented in Figure 2.3a. The fact that of course Newton's method outperform the subgradient scheme is not surprising. Perhaps the surprising fact is that Newton's method outperforms subgradient scheme, despite the fact the computation of the dual Newton step is based on a discrete Poisson equation (essentially diffusion with input).

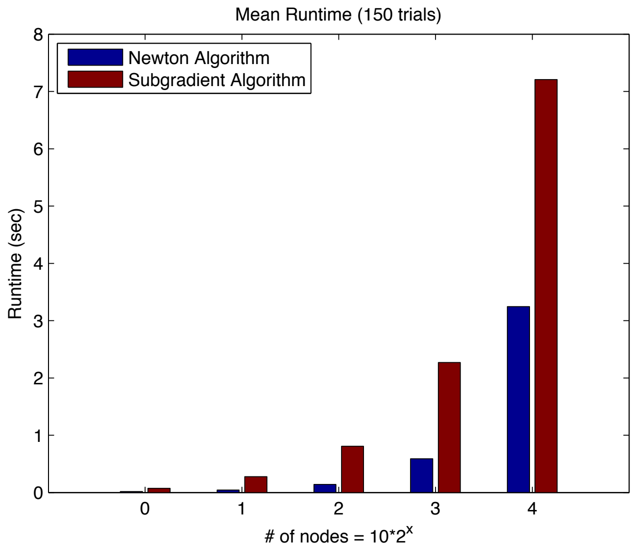
On average the Newton's method terminates in less than half of the subgradient runtime and exhibits a tighter variance. This is a representative sample with respect to varying the number of nodes. As shown in Figure 2.3b, the Newton's method algorithm completes in significantly less time on average for all of the graphs evaluated.

2.3.1 Hidden Costs of Consensus

A completely distributed implementation of Newton's method is the main benefit of the consesus based Newton method proposed. Throughout this chapter, convergence rate is measured by the number of primal-dual iterations required. This measure is misleading because within each primal-dual iteration a consensus process is used, first to compute the direction of descent and then another is used to approximate the backtracking line search. Each of these internal iterations require nodes to communicate with neighboring nodes, significantly increasing the communication overhead of the method. While this trade off is not addressed in Chapter 2, Chapter 3 addresses it directly, taking into account both the primal-dual iterations required for convergence and the communication overhead.



(a) Runtime Histogram, 160 node random networks with mean node degree 5



(b) Average Runtime by number of nodes in the network, 150 samples per network size

Figure 2.3: Newton's Method vs. Subgradient Method

Chapter 3

Accelerated Dual Descent

This chapter develops the Accelerated Dual Descent (ADD) algorithm for solving the minimum convex cost network flow problem using a limited number of local information exchanges while guaranteeing superlinear convergence to a neighborhood of the optimum. The convex minimum cost network flow problem and the study of its dual problem are key building blocks in the study of network optimization.

Minimum convex cost network flow problems can be solved in a distributed manner via dual subgradient descent, [Bertsekas, 1998]. Nodes keep track of variables associated with their outgoing edges and undertake updates based on their local variables and variables available at adjacent nodes. Unfortunately, subgradient descent is known to exhibit prohitively slow convergence in many applications. The natural alternative to accelerate convergence is to use second order Newton methods (see Algorithm 9.2 in [Boyd and Vandenberghe, 2004]), but they cannot be implemented in a distributed manner because matrix inversion is a global operation. As discussed in the previous chapter, a consensus process can be used to approximate the Newton direction but this method requires an additional iterative layer to compute the approximate Newton direction resulting in an arbitrarily large communication requirement between neighboring nodes with each dual update.

Another way to achieve improvements over subgradient methods is to implement Nesterov type accelerated methods summarized in [Tseng, 2008]. These methods work well when the proximal operator has a simple closed form. To be distributed, it is required that the proximal operator be computable using local information, which limits the convergence rates that can be achieved. State of the art proximal gradient methods are compared with ADD in Section 3.5.2. Krylov subspace methods, in particular conjugate gradient descent, can achieve second order convergence when computed centrally, see chapter 6 of [Saad, 2003]. Unfortunately, conjugate gradient and other second order Krylov subspace methods rely heavily on inner products for an orthogonalization procedure which leads to improved convergence rates, see chapter 9 of [Watkins, 2007]. Even in the best case, these inner products violate the communication limitations for this problem, for example see the method proposed in [Klincewicz, 1983].

In this chapter, the underlying structure of the network flow optimization problem is exploited to implement an approximate Newton-type method using only local information. The dual Hessian is a weighted Laplacian of the graph representing our communication network. Using this structure, we can approximate the dual Hessian inverse using local information. Our particular insight is to consider a Taylor's expansion of the inverse Hessian (as in Section 5.8 of [Horn and Johnson, 1985]), which, being a polynomial with the Hessian matrix as variable, can be implemented through local information exchanges. More precisely, considering only the zeroth order term in the Taylor's expansion yields an approximation to the Hessian inverse based immediately available information. The first order approximation requires information available at neighboring nodes and in general, the Nth order approximation necessitates information from nodes located N hops away. The resultant family of ADD algorithms permits a tradeoff between the accuracy of the Hessian approximation and communication cost. We use ADD-N to represent the Nth member of the ADD family which uses information from terminals N hops away.

3.1 Preliminaries

Consider a network represented by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with node set $\mathcal{N} = \{1, \ldots, n\}$, and edge set $\mathcal{E} = \{1, \ldots, E\}$. The network is deployed to support a single information flow specified by incoming rates $b_i > 0$ at source nodes and outgoing rates $b_i < 0$ at sink nodes. Rate requirements are collected in a vector b, which to ensure problem feasibility has to satisfy $\sum_{i=1}^{n} b_i = \mathbf{1}^T b = 0$. Our goal is to determine a flow vector $x = [x_e]_{e\in\mathcal{E}}$, with x_e denoting the amount of flow on edge $e = (i, j)$. Flow conservation is enforced as $Ax = b$, where A the $n \times E$ node-edge incidence matrix defined

$$[A]_{ie} = \begin{cases} 1 & \text{if edge } e \text{ leaves node } i, \\ -1 & \text{if edge } e \text{ enters node } i, \\ 0 & \text{otherwise.} \end{cases}$$

The Algebraic connectivity of $\mathcal{G}$ is the second smallest eigenvalue of the graph Laplacian AA' and d_{max}, the maximum degree of any node in the graph $\mathcal{G}$ is an upper bound on the largest eigenvalue of AA'. We define the penalty as a convex scalar cost function $\phi_e(x_e)$ denoting the cost of x_e units of flow traversing edge e. The convex min-cost flow network optimization problem is then defined as

$$\text{minimize } f(x) = \sum_{e=1}^{E} \phi_e(x_e), \quad \text{subject to: } Ax = b. \tag{3.1}$$

Assumption 3. The Network $\mathcal{G}$ has the following properties:

(a) Connected with algebraic connectivity lower bounded by a constant ω

(b) Non-bipartite

In Assumption 3(a), we quantify the ability of the network to spread information via an algebraic connectivity bound. Assumption 3(b) that $\mathcal{G}$ is non-bipartite guarantees that the normalized Laplacian on the $\mathcal{G}$ has largest eigenvalue strictly upper bounded by 2. Due to our use of λ for dual variables, we use $\mu(X)$ to denote eigenvalues of a matrix X and we order them $|\mu_1| \le |\mu_2| \le \cdots \le |\mu_n|$. Norm notation, $||v||$ and $||X||$ are the Euclidean 2-norm and induced Euclidean 2-norm, respectively. The notation $X \prec Y$ for matrices $X, Y \in \mathbb{S}^n$ indicates that $v'Xv < v'Yv, \forall v \in \mathbb{R}^n$.

Assumption 4. The objective functions $\phi_e(\cdot)$ have the following properties for all e:

(a) Twice continuously differentiable, strongly convex and satisfies $\gamma \le \phi_e''(\cdot) \le \Gamma$

(b) Lipschitz Hessian Inverse $|1/\phi_e''(y) - 1/\phi_e''(\bar{y})| \le \xi|y - \bar{y}|$

Assumption 4 restricts the allowable primal objective functions to those which will yield a dual problem meeting the standard criteria for application of Newton's method, see Chapter 9.5 of [Boyd and Vandenberghe, 2004]. These assumptions are sufficient for convergence, but are not necessary. Restricting to this case allows us to focus on the core mechanisms of a Newton type method in the dual domain. The conditioning assumption on the primal objective can be very limiting as it disallows the negative log function, the minimum delay utility and other potentially interesting objectives. However, this becomes a non-issue when capacity constraints are introduced, because these functions become well conditioned on appropriately bounded domains. Generalization to a capacity constrained variant of this problem is addressed in Chapter 4. For analysis of additional relaxations on the conditions in Assumption 4, the reader is directed to [Kelley, 2003] and [Bonnans et al., 2006].

3.2 Network Optimization

Attention is focused on solutions in the dual domain. Computing the Lagrange dual of a convex minimization with equality constraints yields maximization of a concave function. In the case of (3.1) the Lagrange dual is given by

$$\max_{\lambda} f(x(\lambda)) - \lambda'(Ax(\lambda) - b) \tag{3.2}$$

where the primal optimizers of the Lagrangian are defined

$$x(\lambda) = \arg\min_{x} f(x) - \lambda'(Ax - b). \tag{3.3}$$

Due to the separability of the objective $f(x) = \sum_e \phi_e(x_e)$ and Assumption 4 we can compute the flow on edge $e = (i, j)$ directly from the dual variables at node i and j:

$$x_e(\lambda) = (\phi_e')^{-1}(\lambda_i - \lambda_j). \tag{3.4}$$

Subscript notation, λ_i is used for elements of the vector $\lambda \in \mathbb{R}^n$. For notational convenience the dual is recast,

$$\min_{\lambda} \; q(\lambda) \;\; = \;\; \lambda'(Ax(\lambda) - b) - f(x(\lambda)), \tag{3.5}$$

by minimizing the negation of the objective in (3.2). From this point on we will consider solutions to the dual problem (3.5) and use (3.4) to compute the associated primal optimal variables. To further proceed we outline the consequences of Assumptions 3 and 4 with regards to (3.5).

Lemma 1. The dual objective $q(\lambda) = \lambda'(Ax(\lambda) - b) - f(x(\lambda))$ has the following properties.

(a) The dual Hessian is the weighted Laplacian

$$\nabla^2 q(\lambda) = A[\nabla^2 f(x(\lambda))]^{-1} A',$$

(b) is strongly convex on the subspace $\mathbf{1}^{\perp}$

$$\frac{1}{M} v'v \leq v'\nabla^2 q(\lambda) v \qquad \forall v \in \mathbf{1}^{\perp},$$

with $M = \frac{\Gamma}{\omega}$ where ω and Γ are defined in Assumptions 3 and 4,

(c) has uniformly upper bounded spectrum

$$v'\nabla^2 q(\lambda) v \leq \frac{1}{m} v'v \qquad \forall v \in \mathbb{R}^n,$$

with $m = \frac{\gamma}{2d_{max}}$ where γ is defined in Assumption 4

(d) and is a Lipschitz function of λ, i.e.,

$$||\nabla^2 q(\lambda) - \nabla^2 q(\bar{\lambda})|| \leq L||\lambda - \bar{\lambda}||.$$

where $L = 4d_{max}\xi/\gamma$; ξ and γ are defined in Assumption 4.

Proof :

(a) Consider given dual $\bar{\lambda}$ and primal $\bar{x} = x(\bar{\lambda})$ variables, and consider the second order approximation of the primal objective centered at $\bar{x}$,

$$\hat{f}(y) = f(\bar{x}) + \nabla f(\bar{x})'(y - \bar{x}) + \frac{1}{2}(y - \bar{x})'\nabla^2 f(\bar{x})(y - \bar{x}) \tag{3.6}$$

which is equivalent to approximating each primal objective $\phi_e(\cdot)$ by its local approximation $\hat{\phi}_e(\cdot)$ constructed using the a second order Taylor expansion around the point $\bar{x}_e$. We consider the optimization problem $\max_y -\hat{f}(y)$ subject to $Ay = b$, which is a quadratic maximization approximating (3.1). The dual of the approximate problem is also quadratic,

$$\min_{\lambda \in \mathbb{R}^n} \hat{q}(\lambda) = \min_{\lambda \in \mathbb{R}^n} \frac{1}{2}\lambda' A \nabla^2 f(\bar{x})^{-1} A' \lambda + p'\lambda + r. \tag{3.7}$$

The dual Hessian $\nabla^2 \hat{q}(\lambda) = A[\nabla^2 f(\bar{x})]^{-1} A'$ is computed by differentiating (3.7) twice with respect to λ. By building our approximation at the point $\bar{x} = x(\bar{\lambda})$, the expression (3.7) is exact at the primal dual point $(\bar{x}, \bar{\lambda})$, yielding the dual Hessian $\nabla^2 q(\bar{\lambda}) = A[\nabla^2 f(x(\bar{\lambda}))]^{-1} A'$. ■

(b) From part (a) we have $\nabla^2 q(\lambda) = A[\nabla^2 f(x(\lambda))]^{-1} A'$. We can get the lower bound by choosing $y = A'v$ to correspond with the eigenvector of $\nabla^2 f(x(\lambda))$ with the largest eigenvalue Γ defined in Assumption 4(a). Then $v'H(\lambda)v = y'[\nabla^2 f(x(\lambda))]^{-1} y \geq y'y\frac{1}{\Gamma}$. Since $v \in \mathbf{1}^{\perp}$, $y'y \geq \omega v'v$ thus $M = \Gamma/\omega$ where is ω is the algebraic connectivity defined in Assumption 3(a). ■

(c) We construct the upper bound by condsidering $v'H(\lambda)v = y'[\nabla^2 f(x(\lambda))]^{-1} y \leq \frac{y'y}{\gamma}$ from Assumption 4(a) where $y = A'v$. The bound can be rewritten in terms of v as $\frac{y'y}{\gamma} = \frac{v'(AA')v}{\gamma}$. The relation $\mu_n(AA') \leq 2d_{max}$ is given by Theorem 2(b) in [Mohar and Alavi, 1991] from the fact that AA' is the unweighted graph Laplacian of $\mathcal{G}$, leaving us with final upper bound $v'H(\lambda)v \leq v'v\frac{2d_{max}}{\gamma}$. We let $m = \gamma/(2d_{max})$ completing the proof. ■

(d) From part (a) we have $\nabla^2 q(\lambda) = A[\nabla^2 f(x(\lambda))]^{-1} A'$. Considering a change of coordinates

$y = A'v$ and the definition of the matrix norm as $||X|| = \max_{y \neq 0} ||Xy||/||y||$ we have

$$||\nabla^2 q(\lambda) - \nabla^2 q(\bar{\lambda})|| \leq 2d_{max}||[\nabla^2 f(x(\lambda))]^{-1} - [\nabla^2 f(x(\bar{\lambda}))]^{-1}|| \tag{3.8}$$

because $\mu_n(AA') \leq 2d_{max}$ holds from the fact that AA' is the unweighted graph Laplacian of $\mathcal{G}$. Since $[\nabla^2 f(x(\lambda))]^{-1}$ is diagonal, the matrix norm $||[\nabla^2 f(x(\lambda))]^{-1} - \nabla^2 f(x(\bar{\lambda}))]^{-1}||$ reduces to $\max_e |1/\phi_e''(x(\lambda)) - 1/\phi_e''(x(\bar{\lambda}))|$, which is finite and positive from Assumption 4(a). Applying Assumption 4(b), we have

$$||\nabla^2 q(\lambda) - \nabla^2 q(\bar{\lambda})|| \leq 2d_{max}\xi \max_e ||x_e(\lambda) - x_e(\bar{\lambda})||. \tag{3.9}$$

We can differentiate equation (3.4) using the chain rule,

$$\left|\frac{\partial}{\partial \lambda_s}[\phi_e']^{-1}(\lambda_i - \lambda_j)\right| = \begin{cases} \frac{1}{\phi_e''([\phi_e']^{-1}(\lambda_i - \lambda_j))} & \text{if } s = i \text{ or } s = j \\ 0 & \text{else.} \end{cases} \tag{3.10}$$

Applying the assumption that $\gamma \leq \phi_e''(\cdot) \leq \Gamma$,

$$\left|\frac{\partial}{\partial \lambda_s}[\phi_e']^{-1}(\lambda_i - \lambda_j)\right| \leq \frac{1}{\gamma} \tag{3.11}$$

for all edges. Using the fact that $|\lambda_i - \lambda_j| \leq 2||\lambda||$ and the definition of the derivative,

$$|x_e(\lambda) - x_e(\bar{\lambda})| \leq \frac{|\lambda_i - \lambda_j + \bar{\lambda}_j - \bar{\lambda}_i|}{\gamma} \leq \frac{2||\lambda - \bar{\lambda}||}{\gamma} \qquad \forall e \tag{3.12}$$

Applying (3.12) to (3.9), we have

$$||\nabla^2 q(\lambda) - \nabla^2 q(\bar{\lambda})|| \leq \frac{4d_{max}\xi}{\gamma}||\lambda - \bar{\lambda}|| \tag{3.13}$$

and we conclude that $\nabla^2 q(\lambda)$ is Lipschitz with constant $L = 4d_{max}\xi/\gamma$.

■

Lemma 1 recovers the key assumptions of Newton's Method defined in Section 9.5 of [Boyd and Vandenberghe, 2004], for the problem defined in (3.5). These results are direct consequences of Assumptions 3 and 4.

3.2.1 Gradient Descent

The benchmark distributed solution to (3.1) is the dual subgradient method. In our problem $q(\lambda)$ is differentiable so we have access to the gradient $g(\lambda) = \nabla q(\lambda)$. Consider an iteration index k, an arbitrary initial vector λ_0 and define iterates λ_k generated by the recursion

$$\lambda_{k+1} = \lambda_k - \alpha_k g_k \qquad \text{for all } k \geq 0, \tag{3.14}$$

where $g_k = g(\lambda_k) = \nabla q(\lambda_k)$ denotes the gradient of the dual function $q(\lambda)$ at $\lambda = \lambda_k$. A first important observation is that one can compute the gradient as $g_k = Ax(\lambda_k) - b$ with the vector $x(\lambda_k)$ having components $x_e(\lambda_k)$ as determined by (3.4) with $\lambda = \lambda_k$, as in Section 6.4 of [Bertsekas, 1999]. A second important observation is that because of the sparsity pattern of the node-edge incidence matrix A the ith element $[g_k]_i$ of the gradient g_k can be computed as

$$[g_k]_i = \sum_{e=(i,j)} x_e(\lambda_k) - \sum_{e=(j,i)} x_e(\lambda_k) - b_i \tag{3.15}$$

The algorithm in (3.14)-(3.15) lends itself to distributed implementation. Each node i maintains information about its dual iterates $[\lambda_k]_i$ and primal iterates $x_e(\lambda_k)$ of outgoing edges $e = (i, j)$. Gradient components $[g_k]_i$ are evaluated as per (3.15) using local primal iterates $x_e(\lambda_k)$ for $e = (i, j)$ and primal iterates of neighboring nodes $x_e(\lambda_k)$ for $e = (j, i)$. Dual variables are then updated as per (3.14). We proceed to update primal variables as per (3.4). This update necessitates local multipliers $[\lambda_k]_i$ and neighboring multipliers $[\lambda_k]_j$.

Distributed implementation is appealing because it avoids the cost and fragility of collecting all information at a centralized location. However, the practical applicability of gradient descent is hindered by slow convergence rates; see e.g., [Nedić and Ozdaglar, 2008, 2009a].

3.2.2 Newton's Method

The Newton method is a scaled version of gradient descent. In lieu of (3.14), iterates are given by

$$\lambda_{k+1} = \lambda_k + \alpha_k d_k \qquad \text{for all } k \geq 0, \tag{3.16}$$

where d_k is the Newton direction at iteration k and α_k is a properly selected step size. The Newton direction,

$$H_k d_k = -g_k, \tag{3.17}$$

where $H_k = H(\lambda_k) = \nabla^2 q(\lambda_k)$ is the Hessian of the dual function. From Lemma 1(a) we have

$$H_k = A[\nabla^2 f(x(\lambda_k))]^{-1} A'. \tag{3.18}$$

From the definition of $f(x)$ in (3.1) it follows that the primal Hessian $\nabla^2 f(x_k)$ is a diagonal matrix. From Assumption 1(a), we know $[\nabla^2 f(x_k)]^{-1}$ exists and can be computed locally for each edge. From Lemma 1(a) we know that H_k is a weighted Laplacian of the connected graph $\mathcal{G}$ and thus has rank $n-1$ and zero eigenvalue associate with the eigenvector $\mathbf{1}$. The Newton direction can be represented exactly using the pseudoinverse

$$d_k = -H_k^{\dagger} g_k \tag{3.19}$$

because g_k lies in the subspace $\mathbf{1}^{\perp}$. However, the pseudoinverse $H_k^{\dagger}$ is a dense matrix and computing d_k requires global information. We are therefore interested in approximations of the Newton direction requiring local information only.

3.2.3 Matrix Splitting and the Dual Hessian Psuedoinverse

Matrix splitting is technnique used to shift and scale eigenvalues in order to make iterative methods tractable.

Definition 3.1. Define the **Matrix Splitting** $H_k = D_k - B_k$, where diagonal matrix D_k is constructed

$$[D_k]_{ii} = [H_k]_{ii} \qquad \forall i \in \mathcal{V} \tag{3.20}$$

and the matrix B_k is

$$[B_k]_{ii} = 0 \, \forall i \qquad \text{and} \qquad [B_k]_{ij} = -[H_k]_{ij} \, \forall i, j \neq i. \tag{3.21}$$

Using the splitting from Definition 3.1, a symmetric weighted adjacency matrix with desirable spectral properties can be constructed as follows.

Definition 3.2. Define the **Weighted Adjacency** matrix as

$$F_k = \frac{1}{\delta}\left(B_k + \delta I - D_k\right) \tag{3.22}$$

where D_k and B_k are defined in Definition 3.1 and the constant δ is defined

$$\delta = \frac{4d_{max}}{\gamma} = \frac{2}{m} \tag{3.23}$$

where m and γ are defined in Assumption 4.

Definitions 3.1 and 3.2 allow us to rewrite the Hessian matrix as

$$H_k = \delta(I - F_k). \tag{3.24}$$

This construction is motivated by the properties of lazy random walks. A lazy random is created by taking a random walk matrix P and replacing it with the random walk matrix $1/2(I + P)$. The addition of self loops for all nodes eliminates periodicity giving us nonnegative eigenvalues and stochasticity leads to upper bounding those eigenvalues by 1. For more on the properties of lazy random walks see Chapter 1 in [Chung, 1997]. With our splitting, we do not precisely construct a lazy random walk so we prove stochasticity and characterize the spectral properties of our splitting in the following lemmas.

Lemma 2. The spectrum of the Weighted Adjacency matrix F_k is bounded

$$0 \preceq F_k \preceq I.$$

Proof : From Lemma 1, H_k is a weighted graph Laplacian yielding the following properties by defintion (see [Chung, 1997]),

$$[H_k]_{ii} = -\sum_{j \neq i} [H_k]_{i,j} \geq 0 \qquad \forall i, \tag{3.25}$$

and

$$[H_k]_{ij} \leq 0 \qquad \forall (i,j), j \neq i. \tag{3.26}$$

Combining these Laplacian properties with our Definition 3.2, we have

$$[F_k]_{ii} = \frac{1}{\delta}\left([B_k]_{ii} + \delta - [D_k]_{ii}\right) \qquad \forall i. \tag{3.27}$$

Applying Definition 3.1, $[B_k]_{ii} = 0$, yielding

$$[F_k]_{ii} = \frac{1}{\delta}\left(\delta - [D_k]_{ii}\right) = 1 - \frac{1}{\delta}[H_k]_{ii} \qquad \forall i. \tag{3.28}$$

Similarly considering the off diaginal elements yields

$$[F_k]_{ij} = \frac{1}{\delta}[B_k]_{ij} = -\frac{1}{\delta}[H_k]_{ij} \geq 0 \qquad \forall (i,j), j \neq i. \tag{3.29}$$

Furthermore, from nonegativity in (3.29), the absolute sum of the off-diagonal elements equals the simple sum of the off-diagonal elements

$$\sum_{j \neq i} |[F_k]_{ij}| = \sum_{j \neq i} [F_k]_{ij} = -\frac{1}{\delta}\sum_{j \neq i} [H_k]_{ij} \geq 0. \tag{3.30}$$

In the following argument we apply the Gersgorin disc Theorem, to characterize the eigenvalues of F_k. From the statement of Theorem 6.1.1 in [Horn and Johnson, 1985], the eigenvalues of F_k must lie in the set $\mathcal{Z}$, where

$$\mathcal{Z} = \cup_i \mathcal{D}_i \tag{3.31}$$

and the discs $\mathcal{D}_i$ are defined

$$\mathcal{D}_i = \left\{ z \in \mathbb{C} : |z - [F_k]_{ii}| \leq \sum_{j \neq i} |[F_k]_{ij}| \right\} \qquad \forall i. \tag{3.32}$$

Since F_k is a symmetic matrix, it must have real eigenvalues allowing us to reduce the discs to intervals on the real line,

$$\mathcal{D}_i = \left\{ z \in \mathbb{R} : |z - [F_k]_{ii}| \leq \sum_{j \neq i} |[F_k]_{ij}| \right\} \qquad \forall i. \tag{3.33}$$

Using (3.28) and (3.30), we can rewrite (3.33) as

$$\mathcal{D}_i = \left\{ z \in \mathbb{R} : 1 - \frac{2}{\delta}[H_k]_{ii} \leq z \leq 1 - \frac{1}{\delta}[H_k]_{ii} + \frac{1}{\delta}[H_k]_{ii} \right\} \qquad \forall i \tag{3.34}$$

and using Lemma 1, $[H_k]_{ii} \leq \frac{1}{m} = \frac{\delta}{2}$, we simplify to

$$\mathcal{D}_i \subseteq \{ z \in \mathbb{R} : 0 \leq z \leq 1 \} \qquad \forall i. \tag{3.35}$$

Since (3.35) holds for all discs i, it holds for the union of discs

$$\mathcal{Z} \subseteq \{ z \in \mathbb{R} : 0 \leq z \leq 1 \}, \tag{3.36}$$

guaranteeing that all eigenvalues of F_k lie in the interval $[0, 1]$, completing the proof. ■

Lemma 3. The matrix F_k is doubly stochastic.

Proof : From equation (3.24), we can write

$$F_k = I - \delta^{-1} H_k. \tag{3.37}$$

From Lemma 1, H_k is a graph Laplacian. Therefore, F_k is symmetric and has eigenvalue 1 associated with the vector of all ones. From (3.29) and (3.30), we have elementwise nonnegativity of F_k. Finally from Lemma 1, $[H_k]_{ii} \leq 1/m = \delta/2$ which combined with the Laplacian structure of H_k guarantees $[F_k]_{ij} \leq 1/2 < 1$ for all i, j. ■

Definition 3.3. We compute the **Newton Direction**

$$\begin{aligned} d_k &= \lim_{N \to \infty} d_k^{(N)} && (3.38) \\ &= -\delta^{-1} \sum_{r=0}^{\infty} F_k^r g_k && (3.39) \end{aligned}$$

and the **Hessian Inverse** on $\mathbf{1}^{\perp}$

$$\begin{aligned} H_k^{\dagger} &= \lim_{N \to \infty} \bar{H}_k^{(N)} && (3.40) \\ &= \delta^{-1} \sum_{r=0}^{\infty} F_k^r && (3.41) \end{aligned}$$

which naturally arises from the Neumann expansion of $H_k^{\dagger}$ in light of equation (3.24).

Remark 1. An important consequence of Definitions 3.1, 3.2 and 3.3 is that δ serves as an internal scaling factor in the computation of the Newton direction. This internal scaling is necessary for the asymptotic expression of the Newton direction in (3.41) to converge, which allows for iterative implementation as outlined in Section 3.2.4. Furthermore, δ serves as an explicit scaling factor for the approximate Newton direction, computed in Section 3.3 by truncating (3.41). It is this internal scaling which allows our method to be implemented with relatively large fixed sizes in Section 3.4.1.

3.2.4 Consensus Implementation

Defining a consensus scheme to solve the Newton equation, (3.17) we have the following update

$$d_k^{(r+1)} = F_k d_k^{(r)} - \delta^{-1} g_k \tag{3.42}$$

where the F_k is the weighted adjacency matrix defined in Definition 3.2. In this case we can limit communication costs by iteratively sharing information with 1-hop neighbors. Choosing the initial value to be $d_k^{(0)} = -\delta^{-1} g_k$ results in a sequence of approximations of the Newton direction:

$$\begin{aligned}
d_k^{(0)} &= -\delta^{-1} g_k = -\bar{H}_k^{(0)} g_k \\
d_k^{(1)} &= F_k(-\delta^{-1} g_k) - \delta^{-1} g_k = -\bar{H}_k^{(1)} g_k \\
&\vdots \\
d_k^{(N)} &= -\sum_{r=0}^{N} F_k^r (\delta^{-1} g_k) = -\bar{H}_k^{(N)} g_k
\end{aligned}$$

satisfying the assymptotic condition specified in (3.38) as N approaches infinity. However, one cannot execute the update in (3.42) infinitely for each iteration k. It is therefore necessary to implement a stopping criteria.

3.3 Accelerated Dual Descent

The approximate Newton algorithm is obtained by replacing the Newton step d_k in (3.16) by its approximations $d_k^{(N)} = -\bar{H}_k^{(N)} g_k$. The resultant algorithm is characterized by the iteration

$$\lambda_{k+1} = \lambda_k - \alpha_k \bar{H}_k^{(N)} g_k. \tag{3.43}$$

The choice of N in dictates how much information node i needs from the network to compute the ith element of the approximate Newton direction $d_k^{(N)}$; recall that node i is associated with dual variable $[\lambda_k]_i$.

3.3.1 Local Information Dependence

For the zeroth order approximation $d_k^{(0)}$ only the first term of the sum in (3.38) is considered and it therefore suffices to have access to the local gradient value $[g_k]_i$ to compute the approximate Newton step. The approximation in this case reduces to $d_k^{(0)} = \delta^{-1} g_k$. The first order approximation $d_k^{(1)}$ uses the first two terms of the sum in (3.38) yielding $d_k^{(1)} = \delta^{-1}(I + F_k)\, g_k$. The key observation here is the sparsity pattern of F_k; it is a weighted adjacency matrix so $[F_k]_{ij} = 0$ for all $(i, j) \notin \mathcal{E}$. As a consequence, to compute the ith element of $d_k^{(1)}$ node i needs to collect information that is either locally available or available at nodes that share an edge with i. For the second order approximation $d_k^{(2)}$ we add the term $\delta^{-1} F_k^2 g_k$ to the approximation $d_k^{(1)}$. The sparsity pattern of F_k^2 has nonzero entries matching the 2-hop neighborhoods of each node. Therefore, to compute the ith element of $d_k^{(2)}$ node i requires access to information from neighboring nodes and from neighbors of these neighbors. In general, the Nth order approximation adds a term of the form F_k^N to the $N - 1$st order approximation. The sparsity pattern of F_k^N coincides with the N-hop neighborhood, and computation of the local elements of the Newton step necessitates information from N hops away. We thus interpret (3.38) as a family of approximations indexed by N that yields Hessian approximations requiring information from N-hop neighbors in the network. This family of methods offers an explicit trade off between communication cost and precision of the Newton direction.

3.3.2 Basic Properties

Analysis of the Newton's method takes advantage of the conditioning results from Lemma 1 by using the psuedo-inverse eigenvalue bounds

$$m\, v'v \leq \quad v'H(\lambda)^{\dagger} v \qquad \forall v \in \mathbf{1}^{\perp}, \tag{3.44}$$

$$v'H(\lambda)^{\dagger} v \leq \quad M v'v \qquad \forall v \in \mathbb{R}^n. \tag{3.45}$$

Our algorithm uses $\bar{H}^{(N)}(\lambda)$ in place of $H^{\dagger}(\lambda)$, so we proceed to prove a conditioning bound on our approximation of the Hessian inverse.

Lemma 4. The approximate inverse Hessian remains well conditioned; i.e., for given ADD family index N and dual variable λ it holds that

$$\bar{m}\, v'v \leq v'\bar{H}^{(N)}(\lambda)v \leq \bar{M}\, v'v, \qquad \forall v \in \mathbb{R}^n \tag{3.46}$$

where $\bar{m} = \delta^{-1}$ and $\bar{M} = \delta^{-1}(N+1)$.

Proof : From the definition of $\bar{H}^{(N)}(\lambda)$, as characterized in (3.41), we have

$$v'\bar{H}^{(N)}(\lambda)v = v'\left(\sum_{r=0}^{N} \delta^{-1}F^r\right)v \tag{3.47}$$

where $H := \delta(I - \delta^{-1}F)$, defined in accordance with Defintion 3.24. We form an upper bound using the induced matrix norm,

$$v'\bar{H}^{(N)}(\lambda)v \leq v'v\, \delta^{-1}||\sum_{r=0}^{N} F^r||. \tag{3.48}$$

From Lemma 2, we know that $||F_k^r|| \leq 1$; applying the triangle inequality and counting up the terms yields

$$v'\bar{H}^{(N)}(\lambda)v \leq v'v(N+1)\delta^{-1}, \tag{3.49}$$

establishing the upper bound for all $v \in \mathbb{R}^n$.

Starting with (3.47) and recalling from Lemma 2 that each term in the sum is positive semidefinite; we establish the lower bound

$$v'\bar{H}_k^{(N)}v \geq v'F_k^0 v\delta^{-1} = v'v\delta^{-1}, \qquad \forall v \in \mathbb{R}^n \tag{3.50}$$

by throwing away (necessarily nonnegative) terms $r = 1, \ldots, N$. ■

Lemma 4 guarantees uniform conditioning and strong convexity of our approximate Hessian inverse which are key analytical building blocks for our convergence analysis.

A basic guarantee for any iterative optimization algorithm is to show that it eventually approaches a neighborhood of the optimal solution. This is not immediate for ADD as defined by (3.43) because the errors in the $\bar{H}_k^{(N)}$ approximations to $H_k^{\dagger}$ may be significant. Notwithstanding, it is shown in Lemma 4 that the $\bar{H}_k^{(N)}$ approximations are positive definite for all N and from there it is possible to conclude that the λ_k iterates in (3.43) eventually approach a neighborhood of the optimal λ^*. This claim is summarized in the following theorem, for proof see Proposition A.24 in [Bertsekas, 1999].

Theorem 2. Let λ^* denote the optimal argument of the dual function $q(\lambda)$ of the optimization problem in (3.1) and consider the ADD-N algorithm characterized by iteration (3.43) with $\bar{H}_k^{(N)}$ as in (3.38). Assume $\alpha_k = \alpha$ for all k and that the network graph is not bipartite. Then, for all sufficiently small α,

$$\lim_{k\to\infty} \lambda_k = \lambda^*. \tag{3.51}$$

By continuity of (3.4), convergence of the dual variable to an error neighborhood implies convergence of the primal variables to an error neighborhood. Theorem 2 is the weakest convergence proof presented, but it is included because it serves as a benchmark for algorithm performance. Also, Theorem 2 uses a fixed step size which in many applications of interest is more practical than implementing a line search method.

Definition 3.4. We define the **Newton Error** to be

$$\epsilon_k = H_k d_k^{(N)} + g_k. \tag{3.52}$$

This is a measure of the deviation from true Newton direction because $H_k d_k = -g_k$ [cf. (3.17)] where d_k is the true Newton direction, which is equivalent to having $\epsilon_k = 0$ in (3.52). Therefore, the Newton step approximation error ϵ_k quantifies the deviation of the approximate Newton steps $d_k^{(N)}$ with respect to the actual Newton step d_k.

Definition 3.5. The **connectivity modulus** is defined as the uniform upper bound $\bar{\rho}$,

$$\left|\mu_{n-1}\big(F_k\big)\right| \leq \bar{\rho} \qquad \forall k. \tag{3.53}$$

We can characterize the connectivity modulus by recalling from Lemma 3 that F_k is a stochastic matrix and applying the eigenvalue bound from [Landau and Odlyzko, 1981],

$$|\mu_{n-1}\left(F_k\right)| \leq 1 - \frac{\gamma}{nd_{max}(1 + \text{diam}(\mathcal{G}))} \tag{3.54}$$

where diam(G) is the diameter of the graph $\mathcal{G}$ and d_{max} is the maximal degree of any node in $\mathcal{G}$.

Table 3.1: Network Coeficient for Connectivity Modulus; Examples with $n = 10$ nodes

Network	Diameter	Max Degree	$nd_{max}(1 + \text{diam}(\mathcal{G}))$
Line Graph	9	2	200
Cycle	5	2	120
Cycle w/ crosslinks	3	3	120
Complete Graph	1	9	180
Star Graph	2	9	270
Petersen Graph	2	3	90

In Table 3.1, we exame the effect of the network structure on the connectivity modulus. Based on (3.54) from [Landau and Odlyzko, 1981], the best networks are those which have both small maximum degree and small diameter. Moore graphs, which are $n = 1 + d\sum_{r=0}^{k-1}(d-1)^r$ node regular graphs of degree d with fixed diameter k are nearly optimal with respect to this metric, [Hoffman and R, 1960]. The best example presented, the Petersen Graph is a Ramanujan graph, a well known class of expander graphs, [Murty, 2003]. Expander graphs are noted for rapid mixing in accordance with their spectral characterization; an approach to interpreting the connectivity modulus through spectral bounds is presented [Tutunov et al., 2014]:

$$1 - \frac{2}{\delta\gamma}\sqrt{2\omega} \leq \bar{\rho} \leq 1 - \frac{\omega^2}{8\delta^2\Gamma^2}. \tag{3.55}$$

The bound in (3.55) connects the connectivity modulus directly to the algebraic connectivity, ω of the graph $\mathcal{G}$ as defined in Assumption 3 and provides a lower bound for the connectivity modulus.

Lemma 5. The norms of the Newton approximation errors ϵ_k defined in (3.52) satisfy

$$\|\epsilon_k\| \leq \bar{\rho}^{N+1}\|g_k\|. \tag{3.56}$$

Proof : Combining the definition of ϵ_k with equation (3.41), we have

$$\epsilon_k = -H_k\bar{H}_k^{(N)}g_k + g_k. \tag{3.57}$$

Consider $H_k\bar{H}_k^{(N)}$, applying equation (3.24),

$$H_k\bar{H}_k^{(N)} = \delta(I - F_k)\left(\delta^{-1}\sum_{r=0}^{N} F_k^r\right) = (I - F_k)\sum_{r=0}^{N} F_k \tag{3.58}$$

which is a telescoping sum, simplifying to

$$H_k\bar{H}_k^{(N)} = I - F_k^{N+1}. \tag{3.59}$$

Substituting (3.59) back into (3.57) yields

$$\epsilon_k = F_k^{N+1}g_k. \tag{3.60}$$

The matrix F_k is symmetric and positive definite allowing us to state the eigenvector decomposition

$$F_k = \sum_{i=1}^{n} \mu_i v_i v_i' \tag{3.61}$$

where the vectors v_i form an orthonormal basis and the eigenvalues $0 \leq \mu_i \leq \mu_n$. Since the vectors v_i form an orthonormal basis we have $v_i'v_j = 0$ for all $i \neq j$ and $v_i'v_i = 1$ for all i, allowing us to write

$$F_k^{N+1} = \sum_{i=1}^{n} \mu_i^{N+1} v_i v_i'. \tag{3.62}$$

Since F_k is stochastic, $\mu_n = 1$ with $v_n = \frac{1}{\sqrt{n}}\mathbf{1}$. Consider,

$$||F_k^{N+1} g_k|| \leq \max_{u \in 1^\perp; u \neq 0} \frac{||F_k^{N+1} u||}{||u||} ||g_k|| \tag{3.63}$$

because $g_k \in 1^\perp$. The vector $F_k u$ can be expressed as

$$F_k^{N+1} u = \sum_{i=1}^{n} \mu_i^{N+1} v_i v_i' u \tag{3.64}$$

but $u' v_n = 0$ becuase $v_n \in \text{span}\{\mathbf{1}\}$ yielding

$$F_k^{N+1} u = \sum_{i=1}^{n-1} \mu_i^{N+1} v_i v_i' u. \tag{3.65}$$

Since the eigenvalues are listed in ascending order the maximum in (3.63) is achieved for $u \in \text{span}\{v_{n-1}\}$, yielding

$$||F_k^{N+1} g_k|| \leq \mu_{n-1}^{N+1} \frac{v_{n-1} v_{n-1}' u}{||u||} ||g_k|| = \mu_{n-1}^{N+1} ||g_k||. \tag{3.66}$$

Taking the norm of (3.60) and substituting (3.66) yields

$$||e_k|| \leq \mu_{n-1}^{N+1} ||g_k||; \tag{3.67}$$

applying the definition of $\bar{\rho}$ completes the proof. ■

Lemma 5 establishes the connectivity modulus, $\bar{\rho}$ as a key coefficient capturing the ability of information to spread through the network. Its appearance is natural because the accuracy of our local approximations to the Newton step depend on the network's ability to percolate information. Another key observation about Lemma 5 is that the Newton step approximation error is proportional to the norm of the dual gradient. Another way to interpret Lemma 5 is to observe that the relative Newton error $||\epsilon_k||/||g_k||$ is at worst a constant. Since the dual gradient norm $\|g_k\|$ tends to zero as we approach the dual optimum argument, the approximation error norm $\|\epsilon_k\|$ also approaches zero as iterations progress towards the optimum.

3.4 Convergence Analysis

In this section, we characterize the convergence rate properties of the Accelerated Dual Descent as defined in (3.43). Our results are presented in three phases characterized by ranges of the residual $||g_k||$, modeled after the convergence properties of the Newton method.

3.4.1 Main Results

The first result presented characterizes the phases of convergence of the ADD algorithm with parameter N and the connectivity modulus $\bar{\rho}$. In accordance with analysis of the Newton method, convergence is presented as a global phase with a strict decrease guarantee and a local phase with a stronger convergence rate guarantee. Due to ADD using an approximation of the Netwon direction, the local phase must be divided into two phases: quadratic phase and terminal phase.

Theorem 3. Consider an ADD-N algorithm with iterates λ_k as with approximate Newton step $-\alpha_k H_k^{(N)} g_k$ for given N. The step size $\alpha_k = \alpha$ is selected such that $0 < \alpha \leq \min\left\{1, \frac{2}{(N+1)}\right\}$.

(i) Global Phase. For $||g_k|| > \eta_1$, we have strict decrease in the objective:

$$q(\lambda_{k+1}) < q(\lambda_k) - \frac{\alpha m \eta_1^2}{4}. \tag{3.68}$$

(ii) Quadratic Phase. For $\eta_0 < ||g_k|| \leq \eta_1$, we have quadratic decrease

$$||g_{k+1}|| < \eta_1^{-1}||g_k||^2 \tag{3.69}$$

(iii) Terminal Phase. For $||g_k|| \leq \eta_0$, we have

$$||g_{k+1}|| \leq \hat{p}||g_k||. \tag{3.70}$$

The phase boundaries η_0 and η_1 are defined

$$\eta_0 = \frac{2}{\alpha^2 L \bar{M}^2} (1 - \hat{p}) \hat{p}, \tag{3.71}$$

$$\eta_1 = \frac{2}{\alpha^2 L \bar{M}^2} (1 - \hat{p}), \tag{3.72}$$

with coefficient

$$\hat{p} = \sqrt{1 - \alpha(1 - \bar{\rho}^{N+1})} \in (0, 1). \tag{3.73}$$

The complete proof of Theorem 3 is presented in Sections 3.4.2 and 3.4.3. Section 3.4.2 considers the fixed stepsize selection to prove the existence of the strict decrease phase of Part (i). In Section 3.4.3, the proofs of the quadratic and terminal phases are presented.

Theorem 3 characterizes the convergence of the ADD-N algorithm. A neighborhood around the optimal is characterized by the boundary η_1, outside which strict decrease in the objective is guaranteed. Inside the neighborhood, convergence is quadratic until the neighborhood characterized by η_0 is reached. Within the terminal neighborhood progress is guaranteed to continue at the exponential rate $\hat{p}$.

From Theorem 3, it is clear that ADD-N with small parameter N allows for relatively large fixed stepsizes. In particular, ADD-1 exhibits global convergence with any fixed step size $\alpha \leq 1$. For ADD-0, Theorem 3 also admits the stepsize $\alpha = 1$; this may be suprising at first glance, but due to the internal scaling of ADD, the resulting algorithm is equivalent to gradient descent with a stepsize proportional to $\frac{1}{L}$. Observe $\bar{H}_k^{(0)} = \delta^{-1} I$ and from the constant definitions in Assumption 4, Lemma 1, and Definition 3.2: $\delta \propto L$. It is precisely this internal scaling feature, built into the Newton direction approximation, that allows the ADD algorithm to operate with large fixed stepsizes. Alternatively, this feature can be thought of as a stepsize selection process built into the algorithm.

Remark 2. Theorem 3 tells us that network structure affects the performance of ADD-N algorithms through the connectivity modulus $\bar{\rho}$ because the residual value defining the local convergence neighborhood is bounded by $\bar{\rho}^{N+1}$. When $\bar{\rho}$ is small, $\hat{\rho}$ is small which yields larger η_1. Large η_1 expands the size of the quadratic decrease phase and reduces the constant term in the update expression.

It is fair to say that ADD-N works well in networks with small $\bar{\rho}$ while networks with large $\bar{\rho}$ are difficult in that they require larger N to experience the same rate of convergence. From the bound in (3.54) we can infer topological conditions that effect $\bar{\rho}$. This bound approaches 1 if some of the following situations happen: (i) the number of nodes n is large, (ii) the graph diameter $\text{diam}(G)$ is large, (iii) while not directly obvious from (3.54), the derivation from of (3.54) in [Landau and Odlyzko, 1981] indicates that $\bar{\rho}$ tends toward 1 when then network has large maximum degree, $\Delta(G)$.

Poor performance for large n and large $\text{diam}(G)$ matches the intuition that convergence towards λ^* necessitates propagation of information through the network. Poor performance for large $\Delta(G)$ is counterintuitive but not incongruous with results on consensus on scale free networks, [Newman, 2003]. For a graph of fixed size n the network structure for which the bound in (3.54) is smallest is the one with the minimum product $\Delta(G)\text{diam}(G)$ of maximum degree and diameter. Small world networks [Watts and Strogatz, 1998] have small degree and small diameter by design. It has already being observed that networks which have simultaneously low diameter and low maximum degree have desirable properties in engineered systems, such as fault tolerance and algebraic connectivity, [Olfati-Saber, 2005]. Additional study of the impacts of network connectivity on Accelerated Dual Descent and further characterizations of $\bar{\rho}$ are explored in [Tutunov et al., 2014].

3.4.2 Global Phase Proof for Theorem 3

For the proof of Theorem 3(i), we begin by writing the quadratic representation of $q(\lambda)$ centered at λ_k using the mean value theorem,

$$q(\lambda_{k+1}) = q(\lambda_k) + g_k'(\lambda_{k+1} - \lambda_k) + \frac{1}{2}(\lambda_{k+1} - \lambda_k)'H(z)(\lambda_{k+1} - \lambda_k) \tag{3.74}$$

for $z \in [\lambda_{k+1}, \lambda_k]$. Applying the dual Hessian spectrum bound from Lemma 1 and the definition of the ADD-N update from (3.43) with stepsize α, we have

$$q(\lambda_{k+1}) \le q(\lambda_k) - \alpha g_k' \bar{H}_k^{(N)} g_k + \frac{\alpha^2}{2m} \left\| \bar{H}_k^{(N)} g_k \right\|^2 . \tag{3.75}$$

Since $\bar{H}^{(N)}$ is symmetric and positive definite with maximum eigenvalue $\bar{M}$ from Lemma 4, we have

$$q(\lambda_{k+1}) \le q(\lambda_k) - \alpha g_k' \bar{H}_k^{(N)} g_k + \frac{\alpha^2 \bar{M}}{2m} g_k' \bar{H}_k^{(N)} g_k \tag{3.76}$$

which simplifies to

$$q(\lambda_{k+1}) \le q(\lambda_k) - \alpha \left(1 - \alpha \frac{\bar{M}}{2m} \right) g_k' \bar{H}_k^{(N)} g_k. \tag{3.77}$$

Consider our restriction that $\alpha \le \frac{2}{N+1}$, recalling that $m = 2\bar{m}$ and $\frac{\bar{M}}{\bar{m}} = N+1$, we have $\alpha \le \frac{m}{\bar{M}}$. Applying this relation yields

$$q(\lambda_{k+1}) \le q(\lambda_k) - \frac{\alpha}{2} g_k' \bar{H}_k^{(N)} g_k. \tag{3.78}$$

Applying the lower bound on the spectrum of $\bar{H}_k^{(N)}$ from Lemma 4 and the global phase condition $||g_k|| > \eta_1$, yields

$$q(\lambda_{k+1}) \le q(\lambda_k) - \frac{\alpha \bar{m}}{2} \eta_1^2 \tag{3.79}$$

and observing again that $m = 2\bar{m}$, completes the proof of Theorem 3(i).

3.4.3 Local Phase Proofs for Theorem 3

The result in Theorem 3 (i) guarantees global convergence into any error neighborhood $\|g_k\| \le \eta_1$ of the optimal value because the dual objective is strictly decreasing by, at least, a noninfinitesimal quantity while we remain outside of this neighborhood. In particular, we are guaranteed to reach a point inside the neighborhood $\|g_k\| \le \eta_1$. In order to proceed with the characterization of the the convergence rate inside the neighborhood $\|g_k\| \le \eta_1$, the following property for Lipschitz Hessian functions is derived.

Lemma 6. Consider a twice differentiable objective function $F(y)$ with a Lipschitz continuous Hessian $\nabla^2 F(y)$ with Lipschitz coeficient L. For any y and $\hat{y}$

$$||\nabla F(\hat{y})|| \leq ||\nabla F(y) + \nabla^2 F(y)(y - \hat{y})|| + \frac{L}{2}||y - \hat{y}||^2. \tag{3.80}$$

Proof : We begin with a statement of the fundamental theorem of calculus,

$$\nabla F(\hat{y}) = \nabla F(y) + \int_0^1 \nabla^2 F(y + t(\hat{y} - y))(\hat{y} - y)dt. \tag{3.81}$$

Adding and subtracting the term $\nabla^2 F(y)(y - \hat{y})$, we acquire

$$\nabla F(\hat{y}) = \nabla F(y) + \int_0^1 \nabla^2 \big(F(y + t(\hat{y} - y)) - \nabla^2 F(y)\big)(\hat{y} - y) + \nabla^2 F(y)(y - \hat{y})dt \tag{3.82}$$

which observing that the final term int he integral does not depenf on t can be reduced to

$$\nabla F(\hat{y}) = \nabla F(y) + \nabla^2 F(y)(\hat{y} - y) + \int_0^1 \nabla^2 \big(F(y + t(\hat{y} - y)) - \nabla^2 F(y)\big)(\hat{y} - y)dt. \tag{3.83}$$

Taking the 2-norm, and applying the triangle inequality,

$$\|\nabla F(\hat{y})\| \leq \big\|\nabla F(y) + \nabla^2 F(y)(y - \hat{y})\big\| + \left\|\int_0^1 \nabla^2 \big(F(y + t(\hat{y} - y)) - \nabla^2 F(y)\big)(\hat{y} - y)dt\right\|. \tag{3.84}$$

The norm of the integral is less than the integral of the norm, yielding

$$\|\nabla F(\hat{y})\| \leq \big\|\nabla F(y) + \nabla^2 F(y)(y - \hat{y})\big\| + \int_0^1 \big\|\nabla^2 \big(F(y + t(\hat{y} - y)) - \nabla^2 F(y)\big\| \, \|\hat{y} - y\| \, dt. \tag{3.85}$$

Applying our assumption that the Hessian is Lipschitz, we have

$$\|\nabla F(\hat{y})\| \leq \big\|\nabla F(y) + \nabla^2 F(y)(y - \hat{y})\big\| + \int_0^1 Lt \, \|\hat{y} - y\|^2 \, dt \tag{3.86}$$

and computing the integral completes the proof. ■

Using Lemma 6, the relation that fundamentally characterizes progress in the local phase is proven in the following lemma.

Lemma 7. Consider the dual update in (3.43) with stepsize α_k. The norm of the dual gradient is reduced at each iteration according to the relation

$$||g_{k+1}|| \leq (1 - \alpha_k + \alpha_k \bar{\rho}^{N+1})||g_k|| + \frac{\alpha_k^2 L \bar{M}^2}{2}||g_k||^2 \tag{3.87}$$

where $\bar{\rho}$ is the connectivity modulus defined in (3.54).

Proof : Applying Lemma 6 for the dual objective $q(\lambda)$ for values λ_{k+1} and λ_k, we have

$$||g_{k+1}|| \leq ||g_k + H_k(\lambda_{k+1} - \lambda_k)|| + \frac{L}{2}||\lambda_{k+1} - \lambda_k||^2. \tag{3.88}$$

Substituting our update rule from (3.43),

$$||g_{k+1}|| \leq ||g_k - \alpha_k H_k \bar{H}_k^{(N)} g_k|| + \frac{L}{2}||\alpha_k \bar{H}_k^{(N)} g_k||^2. \tag{3.89}$$

Now consider the term $||I - \alpha_k H_k \bar{H}_k^{(N)}||$, decomposed according to Definition 3.2 and applying equation 3.59,

$$||I - \alpha_k H_k \bar{H}_k^{(N)}|| = ||I - \alpha_k(I - F_k^{N+1})|| \leq 1 - \alpha_k + \alpha_k ||F_k^{N+1}||. \tag{3.90}$$

Applying Definition 3.56,

$$||I - \alpha_k H_k \bar{H}_k^{(N)}|| \leq 1 - \alpha_k + \alpha_k \bar{\rho}^{N+1} \tag{3.91}$$

which substituted into (3.89), yields

$$||g_{k+1}|| \leq (1 - \alpha_k + \alpha_k \bar{\rho}^{N+1})||g_k|| + \frac{L}{2}||\alpha_k \bar{H}_k^{(N)} g_k||^2, \tag{3.92}$$

and applying the upper bound from Lemma 4 completes the proof. ■

Lemma 7 shows that the reduction in the dual gradient at each iteration, (3.87) has a linear term and a quadratic term. The quadratic term is the same term that appears in the analysis of Newton's method, see Chapter 9 in [Boyd and Vandenberghe, 2004] up to multiplication by the coeficient $\hat{p}$. The linear term corresponds to the error in the step approximation due to the truncation of the series in (3.38). Despite the error term, we achieve convergence to the optimal point.

The local phase of the ADD-N algorithm is broken up into a quadratic phase (ii) and a terminal phase (iii). For the proof of part (ii), begin by applying Lemma 7

$$||g_{k+1}|| \leq \hat{p}^2||g_k|| + c||g_k||^2 \tag{3.93}$$

where we denote $c = \frac{\alpha^2 L\bar{M}^2}{2}$ and apply the definition of $\hat{p}$. Applying the lower bound for the quadratic phase, we have

$$||g_{k+1}|| < \left(\frac{\hat{p}^2}{\eta_0} + c\right)||g_k||^2 \tag{3.94}$$

and applying $\eta_0 = c^{-1}\hat{p}(1-\hat{p})$ yields

$$||g_{k+1}|| < c\left(\frac{\hat{p}}{1-\hat{p}} + 1\right)||g_k||^2. \tag{3.95}$$

Observing that $1 = \frac{1-\hat{p}}{1-\hat{p}}$, we simplify to

$$||g_{k+1}|| < \frac{c}{1-\hat{p}}||g_k||^2 \tag{3.96}$$

and recalling $\eta_1 = c^{-1}(1-\hat{p})$ proves our expression for the quadratic phase update.

Proceeding with the proof of Theorem 3(iii), we again consider the statement of Lemma 7 and apply the conditions for the terminal phase, $||g_k|| \leq \eta_0$,

$$||g_{k+1}|| \leq \left(\hat{p}^2 + c\eta_0\right)||g_k||. \tag{3.97}$$

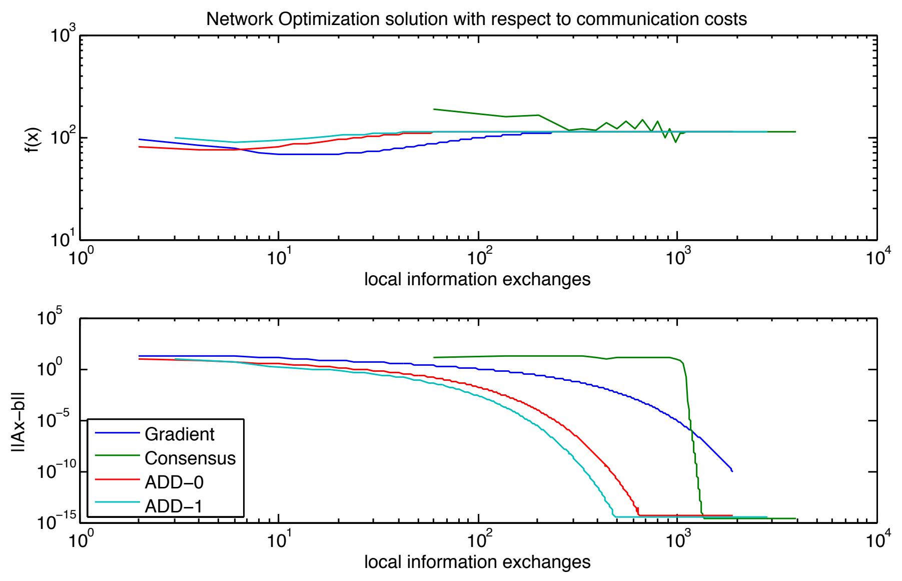


Figure 3.1: Primal objective (top), $f(x_k)$ and primal feasibility (bottom), $\|Ax_k - b\|$ with respect to number of local information exchanges for a sample network optimization problem with 25 nodes and 75 edges.

Applying $\eta_0 = c^{-1}\hat{p}(1-\hat{p})$,

$$||g_{k+1}|| \leq \left(\hat{p}^2 + \hat{p}(1-\hat{p})\right)||g_k|| = \hat{p}||g_k||. \tag{3.98}$$

3.5 Numerical Experiments

Numerical experiments are used to verify the practicality of the ADD-N algorithm. The algorithm is implemented with fixed stepsize $\alpha_k = 1$ for all numerical experiments presented. Our first key result is that the ADD-N algorithm requires significantly fewer iterations to converge than the gradient algorithm, [Nedić and Ozdaglar, 2009b] and can be executed with smaller communication overhead than the consensus based Newton algorithm, [Jadbabaie et al., 2009]. Additionally, the robust routing problem proposed in [Wu et al., 2007] is considered. We demonstrate that the convex network flow framework not solves this problem and that ADD-1 outperforms gradient descent and is competitive with other state of the art methods.

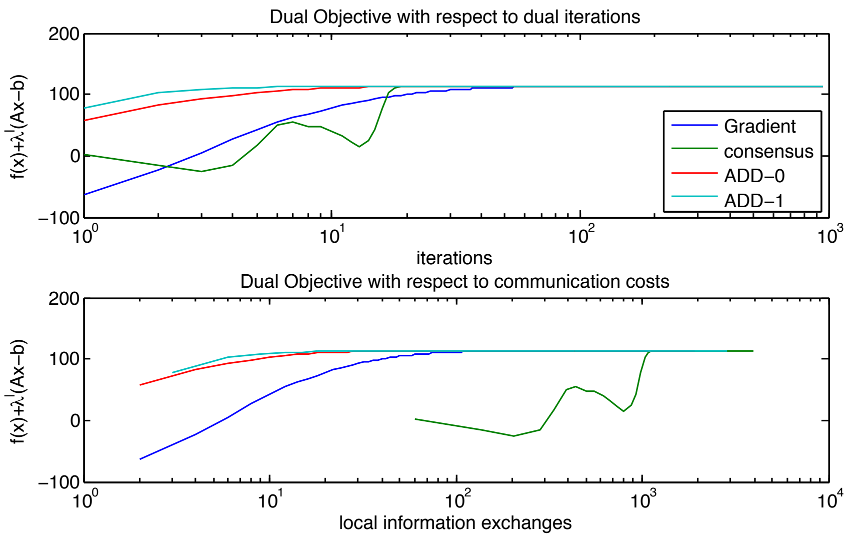


Figure 3.2: Dual objective with respect to dual iterations(top), Dual Objective with respect to communication cost (bottom) for a sample network optimization problem with 25 nodes and 75 edges.

3.5.1 Parameter Selection and Communication Overhead

Figure 3.1 shows convergence metrics in the primal domain for a randomly generated network with 25 nodes and 75 edges. Edges in the network are selected uniformly at random. The flow vector b is chosen to place sources a full diam$(\mathcal{G})$ from the single sink. We use

$$\phi_e(x_e) = \exp(-x_e) + \exp(x_e) \tag{3.99}$$

as our objective function. Communication cost is defined in terms of information exchanges where one *information exchange* is a transmit/receive event where each node i updates any subset of its local copies of variables belonging to its neighbors $j \in \mathcal{N}_i$. Due to our interest limiting to local information, ADD-0 and ADD-1 are studied in Figures 3.1 and 3.2. However, ADD-0 through ADD-3 are studied in the repeated experiments show in Figure 3.4 in order to further characterize the dependence on parameter N.

In Figure 3.2, convergence of the dual objective is demonstrated with respect to both dual iterations and number of local information exchanges required. Different versions of ADD differ

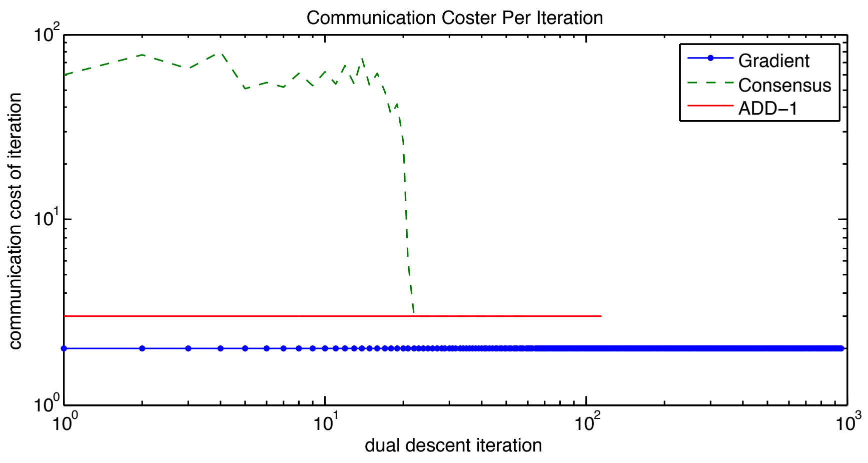


Figure 3.3: The number of local information exchanges required per dual descent iteration is shown for sample network optimization problem with 25 nodes and 75 edges. ADD-N has a fixed local communication requirement per iteration equal to $N + 2$ which yields more consistent convergence rates with respect to communication requirements.

in the number of communication instances required per iteration. When measured in information exchanges ADD-0 and ADD-1 converge at similar rates. Although Consensus based Newton converges in fewer dual iterations that ADD-1, it can be seen in Fig. 3.2 that ADD-1 converges with fewer total information exchanges. In Fig. 3.1, the faster convergence relative to communication cost for ADD-N with smaller N can also be observed in the primal domain.

The communication costs per dual iteration for consensus-based Newton and the gradient descent algorithm also differ from ADD-N. Fig. 3.1 demonstrates the algorithms' progress with respect to the number of times nodes exchange information with their 1-hop neighbors. As shown in Fig. 3.3, ADD-N has a fixed communication cost per iteration while consensus can require arbitrarily many communications– especially during the globally convergent phase during which percise calculation of the Newton direction has limited value. A major benefit of the ADD family is that unlike the consensus based Newton algorithm, precious communication resources are not wasted computing an extremely accurate Newton direction in early iterations when the return on this investment is minimal. Fig. 3.3 also shows that once the consensus Newton algorithm reaches its local phase it requires a very small number of dual iterations to remain below the error threshold. In Figurs 3.1 and 3.2, it is observed that the consensus based Newton method is able to *eventually*

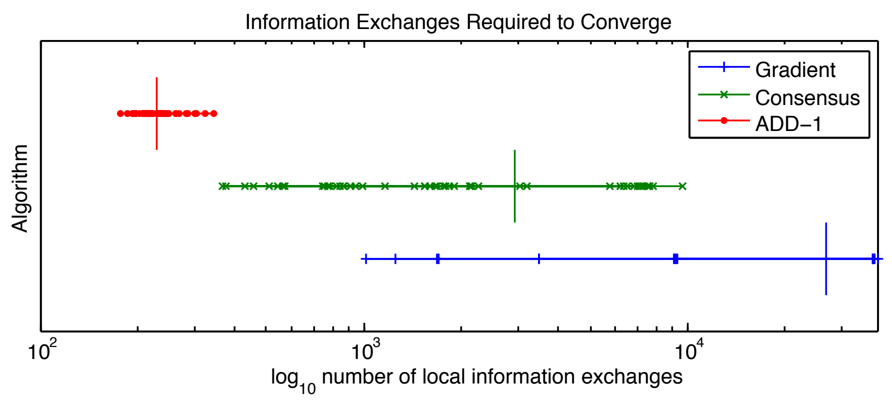


Figure 3.4: The number of local communications required to reach $\|g(\lambda_k)\| \leq 10^{-10}$ for gradient descent, consensus-based Newton and ADD-1 for 50 trials of the network optimization problem on random graphs with 25 nodes and 75 edges. Markers indicate distribution and vertical lines indicate the mean.

achieve a faster convergence rate but communication effort wasted in reaching that phase causes ADD-N to perform better overall.

While Figures 3.1–3.3, indicate a single trial demonstration, Figure 3.4 demonstrates performance over 50 trials of the network optimization problem on random graphs with 25 nodes and 75 edges. Figure 3.4 show that ADD-1 converges (to precision $\|g(\lambda_k)\| \leq 10^{-10}$) faster and with much more consistency than gradient descent or consensus-based Newton. Though increasing N in ADD decreases the number of iterations required, there is not a strict decrease in the number of communications; there is an inherent trade off between spending communication instances to refine the Newton step $d_k^{(N)}$ versus spending communicate to apply the dual update, (3.43). This phenomenon is visible in Fig. 3.5. These experiments are on random graphs of three sizes, small: 25 nodes and 75 edges, medium: 50 nodes and 350 edges and large: 100 nodes and 1000 edges. The flow vector b is selected by placing a source and a sink at diam$(\mathcal{G})$ away from each other. Convergence is defined when residual $\|g_k\| \leq 10^{-10}$ is achieved.

Fig. 3.5 summarizes the comparison between the ADD family and existing methods. Using ADD-N for $N = 0$ or $N = 1$, requires less communication overall than ADD-N for $N = 2$ or $N = 3$; all ADD methods converge on average an order of magnitiude faster than the other

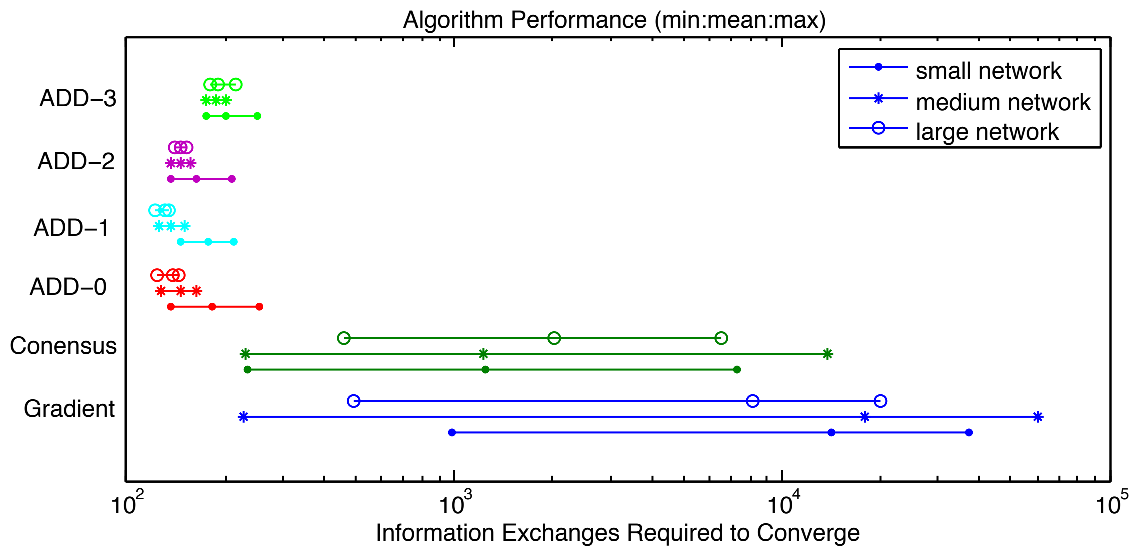


Figure 3.5: Min, mean and max number of local communications required to reach $\|g(\lambda_k)\| \leq 10^{-10}$ for gradient descent, consensus-based Newton and ADD, computed for 35 trials each on random graphs with 25 nodes and 75 edges(small), 50 nodes and 350 edges(medium), and 100 nodes and 1000 edges (large).

algorithms. Consensus-based Newton achieves the rates of ADD on some trials, but fails to do so consistently. The consensus-based Newton is inconsistent due to the wasted communication resources demonstrated by the dual iterations per primal iteration peak shown in Fig. 3.3. This inconsistency comes primarily from variations in the number of dual iterations required to approximate the Newton direction. Thus, the main benefit of ADD-N is consistency as a result of the fixed communication cost per iteration. The behavior of ADD is also explored for graphs of varying size and degree in Fig. 3.5. As the graph size increases the performance gap between ADD and competing methods increases. Consistency of ADD is also apparent since the maximum, minimum, and average information exchanges required to solve (3.1) for different network realizations are similar. This is not the case for consensus-based Newton or for gradient descent. Interestingly, the size of the problem does not significantly impact the performace of the algorithms. This is likely because the crucial factor is the connectivity of the networks. Due to our choice of random graph model and the fact node (squared) to edge ratios remain relatively consistent between network sizes, the connectivities remain comparable.

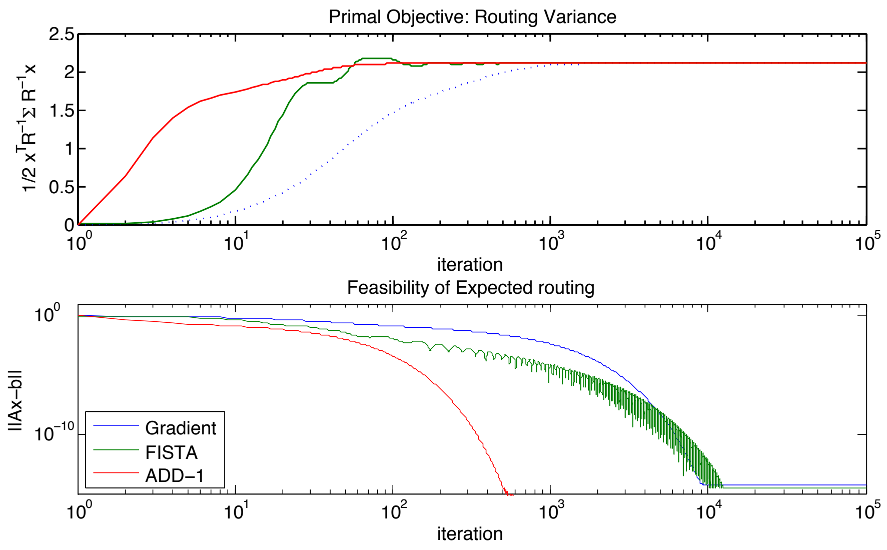


Figure 3.6: The robust routing problem is solved efficiently by the ADD-1 algorithm, while the gradient descent method and FISTA method both require 10s of thousands of iterations. The top figure shows the total variance of the routing selected at iteration t. The bottom figure shows whether the current routing is feasible.

3.5.2 The Robust Routing Problem

The robust routing problem is a network optimization problem focusing on selecting an optimal routing when links have uncertainties in their channel capacity, [Wu et al., 2007]. Each edge has a known variance σ_e and expected capacity r_e. The objective is to select a the flow variable $T_e \in [0, 1]$ that satisfies the conservation of constraint with the minimum total variance, when the expected flow arrivals are given by the vector b.

$$\min_{T_e \in [0,1]} \sum_e \sigma_e T_e^2 \qquad \text{s.t. } ART = b \tag{3.100}$$

We can recover the network flow problem by taking the local coordinate transform $x_e = R_e T_e$. Defining the diagonal matrices $\Sigma = \text{diag}(\sigma_e)$ and $R = \text{diag}(R_e)$ we can state the robust routing problem:

$$\min_{x_e \in [0,R_e]} x' R^{-1} \Sigma R^{-1} x \qquad \text{s.t. } Ax = b \tag{3.101}$$

We solve (3.101) by applying the ADD-1 algorithm and projecting the primal variables onto the interval $[0, R_e]$ during each primal update. While the effect of this projection is not considered analytically in this work, the effect of capacity constraints is discussed at length in [Zargham et al., 2013b]. Figure 3.6 shows a sample solution to (3.101) on a proximity network with 50 nodes, 224 edges. The matrix R is diagonal with values selected uniformly random on [0,1]. The diagonal matrix Σ has nonzero elements Σ_{ii} selected uniformly random on [0,10]. The vector b has a single sink with all other nodes being sources. This example emulates a wireless sensor network streaming data to a base station.

In our example, ADD-1 is compared to gradient descent and the fast iterative shrinking threshold algorithm (FISTA) presented in [Beck and Teboulle, 2009a] and updated for distributed implementation in [Chen and Ozdaglar, 2012]. Distributed FISTA represents the state of the art for fast distributed gradient methods, when methods requiring network-wide message passing are excluded. In figure 3.6, FISTA is a significant improvement over gradient descent but it still requires an order of magnitude more iterations to reach feasibility $||Ax - b|| \leq 10^{-4}$ than ADD-1. The experiment is repeated 100 times and the progress of each algorithm is examined as a histogram after 100 and 1000 iterations; see Figure 3.7. After 100 iterations ADD-1 had reached $||Ax - b|| \leq 10^{-4}$ for nearly 20% of the trials while neither gradient nor FISTA had reached that threshold for any trials. After 1000 iterations ADD-1 had reached feasibility with 60% at machine precision and over 95% smaller than 10^{-4}. Furthermore, relative error in the objective also had over 60% at machine precision and over 95% smaller than 10^{-4}. As with the ADD-1 method the error in the objective is on the same order as the feasibility, 10^{-5}. These experiments demonstrate that while FISTA significantly accelerates gradient descent, the ADD-1 algorithm still consistently solves the robust routing problem an order of magnitude faster than FISTA.

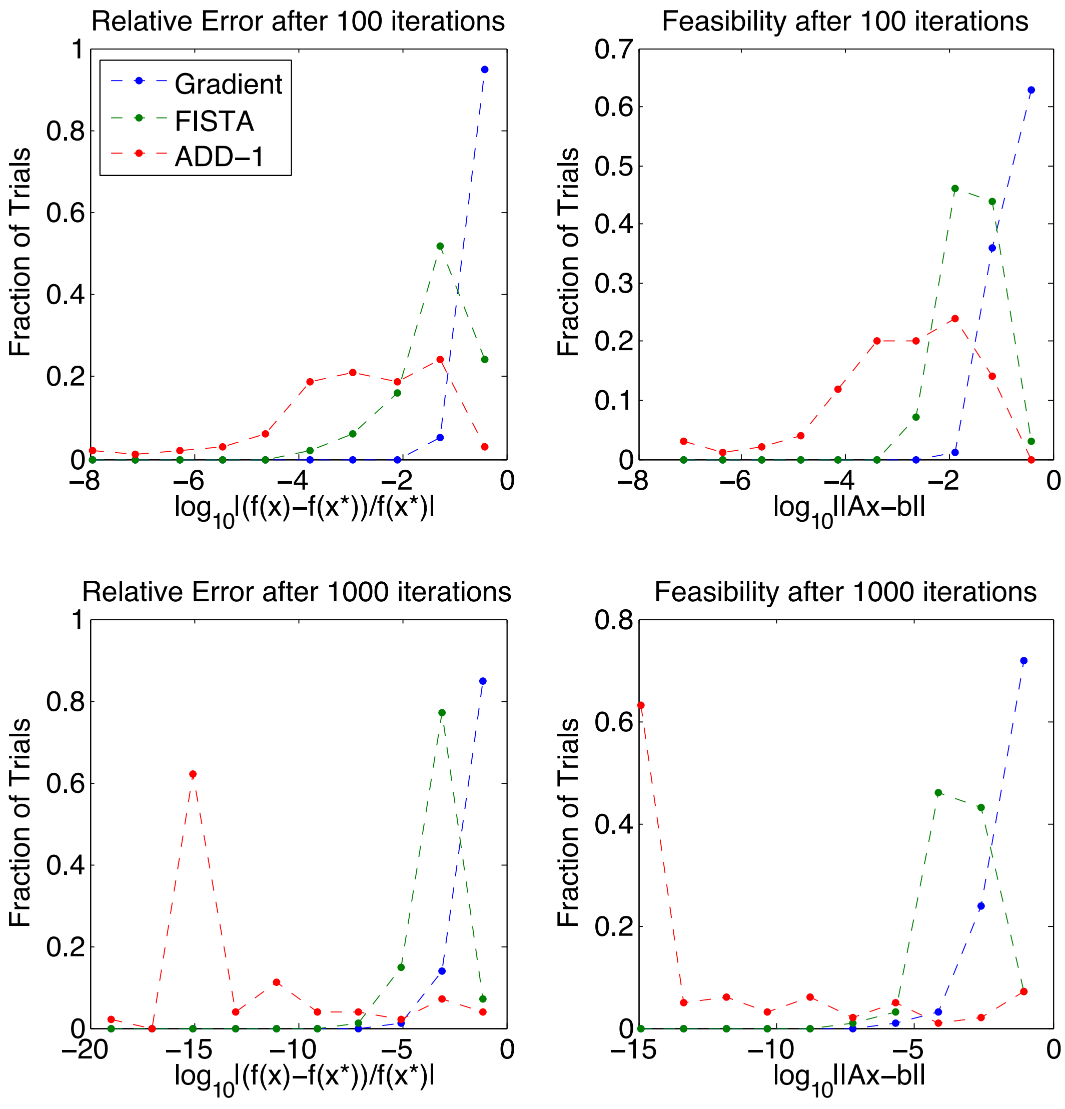


Figure 3.7: Given 100 hundred trials of the robust routing problem, we find that after 100 iterations ADD-1 makes significantly more progress towards the optimal point than FISTA and gradient descent which have high values of infeasibility. After 1000 iterations ADD-1 was reached the optimal to machine precision in more than 60% of the trials while gradient descent and FISTA have at best a feasibility threshold of 10^{-5} and significant errors remaining in the objective value.

Chapter 4

Accelerated Backpressure

Chapter 4 considers the problem of joint routing and scheduling in networks. Packets are accepted from upper layers as they are generated and marked for delivery to intended destinations. To accomplish delivery of information, nodes need to determine routes and schedules capable of accommodating the generated traffic. From a node-centric point of view, individual nodes handle packets that are generated locally as well as packets received from neighboring nodes. The goal of each node is to determine suitable next hops for each flow conducive to successful packet delivery.

A joint solution to this routing and scheduling problem is offered by the Backpressure algorithm [Tassiulas and Ephremides, 1992]. In Backpressure nodes keep track of the number of packets in their local queues for each flow and share this information with neighboring agents. Nodes compute the differences between the number of packets in their queues and the number of packets in neighboring queues for all flows and assign the transmission capacity of the link to the flow with the largest queue differential. The term *backpressure* is used because the algorithm emulates the physical behaviors of fluids under pressure. Regardless of interpretation and despite its simplicity, Backpressure can be proved to be an optimal policy in the following sense: if given arrival rates can be supported by some routing-scheduling policy, they can be supported by Backpressure. Notice that Backpressure relies on queue lengths only and does not need knowledge or estimation of packet arrival rates. It is also important to recognize that Backpressure is an iterative algorithm. Packets are initially routed more or less at random but as queues build throughout the network suitable routes and schedules are eventually learned.

The main drawback of Backpressure is the slow convergence rate of this iterative process for route discovery. This is better understood through an alternative interpretation of Backpressure as a dual stochastic subgradient descent algorithm [Georgiadis et al., 2006; Neely et al., 2005]. Joint routing and scheduling can be formulated as the determination of per-flow routing variables that satisfy link capacity and flow conservation constraints. In this model the packet transmission rates are abstracted as continuous variables. To solve the resulting feasibility problem we associate Lagrange multipliers with the flow conservation constraints and proceed to implement subgradient descent in the dual domain. This solution methodology leads to distributed implementations and for that reason is commonly utilized to find optimal operating points of wired [Eryilmaz and Srikant, 2006; Kelly et al., 1998; Low and Lapsley, 1999] and wireless communication networks [Chiang et al., 2007; Neely, 2006; Ribeiro and Giannakis, 2010]. More important to the discussion here, the resulting algorithm turns out to be equivalent to Backpressure with queue lengths taking the place of noisy versions of the corresponding dual variables. The slow convergence rate of Backpressure is thus expected because the convergence rate of subgradient descent is of order $O(1/\sqrt{t})$ when measured with respect to the expected objective suboptimality of t^{th} iterate, [Nedić and Ozdaglar, 2009b]. Simple modifications can speed up the convergence rate of Backpressure by rendering it equivalent to stochastic gradient descent [Ribeiro, 2009b] which has a convergence rate of order $O(1/t)$ for the expected suboptimality of t^{th} iterate.

Accelerated Backpressure is derived by formulating the packet routing problem as an optimization and applying a variant of the Accelerated Dual Descent method presented in Chapter 3. Unlike Backpressure and Soft Backpressure, the dual variables and the queue lengths do not coincide. The dual variables, referred to as queue priorities converge leading to static routing policy which stabilizes the queues by ensuring that the total packets flowing into the network is balanced by the packets reaching their destinations. Formal proof that the queues are always eventually empty is presented; a decaying stepsize is used for this result which is consistent with stochastic optimization literature. Numerical experiments consider a fixed unit stepsize and demonstrate fast convergence times and small steady state queues relative to Backpressure and Soft Backpressure.

4.1 Backpressure and Queue Stabilization

Consider a given network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of links between nodes. Denote as C_{ij} the capacity of link $(i, j) \in \mathcal{E}$ and define the neighborhood of i as the set $n_i = \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$ of nodes j that can communicate directly with i. There is also a set of information flows $\mathcal{K}$ with the destination of flow $k \in \mathcal{K}$ being the node $o_k \in \mathcal{V}$.

At time index t terminal $i \neq o_k$ generates a random number $a_i^k(t)$ packets to be delivered to o_k. The random variables $a_i^k(t) \geq 0$ are assumed independent and identically distributed across time with expected value $\mathbb{E}\left[a_i^k(t)\right] = a_i^k$. In the same time slot node i routes $r_{ij}^k(t) \geq 0$ units of information through neighboring node $j \in n_i$ and receives $r_{ji}^k(t) \geq 0$ packets from neighbor j.

Algorithm 1: Backpressure at node i

```
1 for $t = 0, 1, 2, \cdots$ do
2     Observe local queue lengths $\{q_i^k(t)\}_k$ for all flows $k$
3     for all neighbors $j \in n_i$ do
4         Send queue lengths $\{q_i^k(t)\}_k$ – Receive $\{q_j^k(t)\}_k$
5         Determine flow with largest pressure:
              $$k_{ij}^* = \underset{k}{\operatorname{argmax}} \left[q_i^k(t) - q_j^k(t)\right]^+$$
6         Set routing variables to $r_{ij}^k(t) = 0$ for all $k \neq k_{ij}^*$ and
              $$r_{ij}^{k_{ij}^*}(t) = C_{ij}\mathbb{I}\left\{q_i^{k_{ij}^*}(t) - q_j^{k_{ij}^*}(t) > 0\right\}$$
          Transmit $r_{ij}^{k_{ij}^*}(t)$ packets for flow $k_{ij}^*$
7     end
8 end
```

The difference between the total number of received packets $a_i^k(t) + \sum_{j \in n_i} r_{ji}^k(t)$ and the sum of transmitted packets $\sum_{j \in n_i} r_{ij}^k(t)$ is added to the local queue – or subtracted if this quantity is negative. Therefore, the number $q_i^k(t)$ of k-flow packets queued at node i evolves according to

$$q_i^k(t+1) = \left[q_i^k(t) + a_i^k(t) + \sum_{j \in n_i} r_{ji}^k(t) - r_{ij}^k(t) \right]^+, \tag{4.1}$$

where the projection $[\cdot]^+$ into the nonnegative reals is because the number of packets in queue cannot become negative. We remark that (4.1) is stated for all nodes $i \neq o_k$ because packets routed to their destinations are removed from the system.

To ensure packet delivery it is sufficient to guarantee that all queues $q_i^k(t+1)$ remain stable. In turn, this can be guaranteed if the average rate at which packets exit queues is less than the rate at which packets are loaded into them. To state this formally observe that the time average limit of arrivals satisfies $\lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t} a_i^k(\tau) = \mathbb{E}\left[a_i^k(t)\right] := a_i^k$ and define the ergodic limit $r_{ij}^k := \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t} r_{ij}^k(\tau)$. If the processes $r_{ij}^k(t)$ controlling the movement of information through the network are asymptotically stationary, queue stability follows if

$$\sum_{j \in n_i} r_{ij}^k - r_{ji}^k \geq a_i^k + \xi \quad \forall\, k, i \neq o_k \tag{4.2}$$

where $\xi > 0$ is a small constant. For future reference, define the vector $r := \{r_{ij}^k\}_{k, i \neq o_k, j}$, built by stacking all the routing variables r_{ij}^k. Since at most C_{ij} packets can be transmitted by the link (i, j) the routing variables $r_{ij}^k(t)$ always satisfy

$$\sum_k r_{ij}^k(t) \leq C_{ij} \tag{4.3}$$

We emphasize that (4.3) holds for all times whereas (4.2) holds in terms of time averages. The joint routing and scheduling problem can be formally stated as the determination of nonnegative variables $r_{ij}^k(t) \geq 0$ that satisfy (4.3) for all times t and whose time average limits r_{ij}^k satisfy (4.2). The Backpressure algorithm solves this problem by assigning all the capacity of the link (i, j) to the flow with the largest queue differential $q_i^k(t) - q_j^k(t)$. Specifically, for each link we determine the flow pressure

$$k_{ij}^* = \underset{k}{\operatorname{argmax}} \left[q_i^k(t) - q_j^k(t) \right]^+ . \tag{4.4}$$

If the maximum pressure $\max_k \left[q_i^k(t) - q_j^k(t) \right]^+ > 0$ is strictly positive we set $r_{ij}^k(t) = C_{ij}$ for $k = k_{ij}^*$. Otherwise the link remains idle during the time frame. The resulting algorithm is summarized in Algorithm 1. The Backpressure algorithm works by observing the queue differentials on each link and then assigning the capacity for each link to the data type with the largest positive queue differential, thus driving the time average of the queue differentials to zero and stabilizing the queues as a consequence. For the generalizations developed in this paper it is necessary to reinterpret Backpressure as a dual stochastic subgradient descent as we do in the following section.

4.1.1 Dual Stochastic Subgradient Descent

Since the parameters that are important for queue stability are the time averages r_{ij}^k of the routing variables $r_{ij}^k(t)$ an alternative view of the joint routing and scheduling problem is the determination of variables r_{ij}^k satisfying (4.2) and (4.3). This can be formulated as the solution of an optimization problem. Let $f_{ij}^k(r_{ij}^k)$ be arbitrary concave functions on $\mathbb{R}_+$ and consider the optimization problem

$$\begin{aligned} r^* := \operatorname{argmax} \quad & \sum_{k, i \neq o_k, j} f_{ij}^k(r_{ij}^k) \\ \text{s.t.} \quad & \sum_{j \in n_i} r_{ij}^k - r_{ji}^k \geq a_i^k + \xi, \quad \forall\, k, i \neq o_k, \\ & \sum_{k \in \mathcal{K}} r_{ij}^k \leq C_{ij}, \qquad\qquad \forall\, (i,j) \in \mathcal{E}. \end{aligned} \tag{4.5}$$

where the optimization is over nonnegative variables $r_{ij}^k \geq 0$. Since only feasibility is important for queue stability, solutions to (4.5) ensure stable queues irrespectively of the objective functions $f_{ij}^k(r_{ij}^k)$.

Since the problem in (4.5) is concave, it has null duality gap and can be solved in dual domain using a descent algorithm. We associate the multipliers λ_i^k with the constraints $\sum_{j \in n_i} r_{ij}^k - r_{ji}^k \geq a_i^k + \xi$ and keep the constraints $\sum_k r_{ij}^k \leq C_{ij}$ implicit. The Lagrangian associated with the optimization problem in (4.5) is then defined as

$$\mathcal{L}(r,\lambda) = \sum_{k,i\neq o_k,j} f_{ij}^k(r_{ij}^k) + \sum_{k,i\neq o_k} \lambda_i^k \bigg(\sum_{j\in n_i} r_{ij}^k - r_{ji}^k - a_i^k - \xi \bigg), \tag{4.6}$$

where we have introduced the vector $\lambda := \{\lambda_i\}_{i=1}^n$ stacking the n local multipliers $\lambda_i := \{\lambda_i^k\}_{k:i\neq o_k}$ which in turn stack all the multipliers at node i that are associated with different flows. The Lagrange dual function is defined as

$$h(\lambda) := \max_{\sum_k r_{ij}^k \leq C_{ij}} \mathcal{L}(r,\lambda). \tag{4.7}$$

To compute a descent direction for $h(\lambda)$ define the primal Lagrangian maximizers as a function of the dual variables,

$$R_{ij}^k(\lambda) := \operatorname*{argmax}_{\sum_k r_{ij}^k \leq C_{ij}} \mathcal{L}(r,\lambda). \tag{4.8}$$

A descent direction for the dual function is available in the form of the dual subgradient whose components $\bar{g}_i^k(\lambda)$ are obtained by evaluating the constraint slack associated with the Lagrangian maximizers

$$\bar{g}_i^k(\lambda) := \sum_{j\in n_i} R_{ij}^k(\lambda) - R_{ji}^k(\lambda) - a_i^k - \xi. \tag{4.9}$$

Since the Lagrangian $\mathcal{L}(r,\lambda)$ in (4.6) is separable in the routing functions $R_{ij}^k(\lambda)$ the determination of the maximizers $R_{ij}^k(\lambda) := \operatorname{argmax}_{\sum_k r_{ij}^k \leq C_{ij}} \mathcal{L}(r,\lambda)$ can be decomposed into the maximization of separate summands. Considering the coupling constraints $\sum_k r_{ij}^k \leq C_{ij}$ it suffices to consider variables $\{r_{ij}^k\}_k$ for all flows across a given link. After reordering terms it follows that we can compute the routing on each edge (i,j),

$$\begin{aligned} R_{ij}^k(\lambda) = \operatorname{argmax} & \sum_k f_{ij}^k(r_{ij}^k) + r_{ij}^k\big(\lambda_i^k - \lambda_j^k\big) \\ \text{s.t.} \qquad & \sum_{k\in\mathcal{K}} r_{ij}^k \leq C_{ij}. \end{aligned} \tag{4.10}$$

Introducing a time index t, subgradients $\bar{g}_i^k(\lambda(t))$ could be computed using (4.9) with Lagrangian maximizers $R_{ij}^k(\lambda(t))$ given by (4.10). A subgradient descent iteration could then be defined to find the variables r^* that solve (4.5); see e.g., [Ribeiro and Giannakis, 2007].

The problem in computing $\bar{g}_i^k(\lambda)$ is that we don't know the average arrival rates a_i^k. We do observe, however, the instantaneous rates $a_i^k(t)$ that are known to satisfy $\mathbb{E}\left[a_i^k(t)\right] = a_i^k$. Therefore,

$$g_i^k(\lambda, t) := \sum_{j \in n_i} R_{ij}^k(\lambda) - R_{ji}^k(\lambda) - a_i^k(t) - \xi, \tag{4.11}$$

is a stochastic subgradient of the dual function in the sense that its expected value $\mathbb{E}\left[g_i^k(\lambda, t)\right] = \bar{g}_i^k(\lambda)$ is the subgradient defined in (4.9). We can then minimize the dual function using a stochastic subgradient descent algorithm. At time t we have multipliers $\lambda(t)$ and determine Lagrangian maximizers $r_{ij}^k(t) = R_{ij}^k(\lambda(t))$ as per (4.10). We then proceed to determine the stochastic subgradient $g_i^k(t) = g_i^k(\lambda(t), t)$ using (4.11) and update multipliers along the stochastic subgradient direction,

$$\lambda_i^k(t+1) = \left[\lambda_i^k(t) - \alpha_t g_i^k(\lambda(t), t)\right]^+ = \left[\lambda_i^k(t) - \alpha_t g_i^k(t)\right]^+, \tag{4.12}$$

where the projection $[\cdot]^+$ into the nonnegatives is to ensure the dual variables stay feasible and α_t is a stepsize sequence appropriately chosen so as to ensure convergence; see e.g., [Ribeiro, 2009b]. For future reference define the local vector $g_i(t) := \{g_i^k(t)\}_{k:i \neq o_k}$ that stacks all the stochastic subgradients at node i that are associated with different flows and the global vector $g(t) := \{g_i(t)\}_{i=1}^n$ that stacks the n local stochastic subgradient vectors $g_i(t)$.

Substituting $g_i^k(t)$ for its explicit expression in (4.11) with $r_{ij}^k(t) = R_{ij}^k(\lambda(t))$ yields the dual variable update

$$\lambda_i^k(t+1) = \left[\lambda_i^k(t) - \alpha_t \left(\sum_{j \in n_i} r_{ij}^k(t) - r_{ji}^k(t) - a_i^k(t) - \xi \right)\right]^+, \tag{4.13}$$

which, except for the presence of the step size α_t and the constant ξ, is equivalent to the queue update in (4.1). Properties of the descent algorithm in (4.13) vary with the selection of the functions $f_{ij}^k(r_{ij}^k)$. Two cases of interest are when $f_{ij}^k(r_{ij}^k) = 0$ and when $f_{ij}^k(r_{ij}^k)$ are continuously differentiable, strongly concave, and monotone decreasing on $\mathbb{R}_+$ but otherwise arbitrary. The former allows an analogy with the backpressure as given by Algorithm 1 while the latter leads to a variation termed Soft Backpressure.

Algorithm 2: Dual stochastic subgradient descent

```
1 for $t = 0, 1, 2, \cdots$ do
2     for all neighbors $j \in n(i)$ do
3         Send duals $\{\lambda_i^k(t)\}_k$ – Receive duals $\{\lambda_j^k(t)\}_k$
4         Determine flow with largest dual variable differential:
              $$k_{ij}^* = \underset{k}{\operatorname{argmax}} \left[\lambda_i^k(t) - \lambda_j^k(t)\right]^+$$
5         Set routing variables to $r_{ij}^k(t) = 0$ for all $k \neq k_{ij}^*$ and
              $$r_{ij}^{k_{ij}^*}(t) = C_{ij}\mathbb{I}\left\{\lambda_i^{k_{ij}^*}(t) - \lambda_j^{k_{ij}^*}(t) > 0\right\}$$
          Transmit $r_{ij}^{k_{ij}^*}(t)$ packets for flow $k_{ij}^*$
6     end
7     Send variables $\{r_{ij}^k(t)\}_{kj}$ – Receive variables $\{r_{ji}^k(t)\}_{kj}$
8     Update multipliers $\{\lambda_i^k(t)\}_k$ along stochastic subgradient
          $$\lambda_i^k(t+1) = \left[\lambda_i^k(t) - \alpha_t\left(\sum_{j\in n(i)} r_{ij}^k(t) + r_{ji}^k(t) - a_i^k(t) - \xi\right)\right]^+$$
9 end
```

4.1.2 Backpressure as Stochastic Subgradient Descent

The classical Backpressure algorithm, [Tassiulas and Ephremides, 1992] can be recovered by setting $f_{ij}^k(r_{ij}^k) = 0$ for all links flows k and links (i, j). With this selection the objective to be maximized in (4.10) becomes $\sum_k r_{ij}^k \left(\lambda_i^k(t) - \lambda_j^k(t)\right)$. To solve this maximization it suffices to find the flow with the largest dual variable differential

$$k_{ij}^* = \underset{k}{\operatorname{argmax}} \left[\lambda_i^k(t) - \lambda_j^k(t)\right]^+ . \tag{4.14}$$

If the value of the corresponding maximum is nonpositive, i.e., $\max_k \left[\lambda_i^k(t) - \lambda_j^k(t)\right]^+ \leq 0$, all summands in (4.10) are nonpositive and the largest objective in (4.10) is attained by making $r_{ij}^k(t) = 0$ for all flows k. Otherwise, since the sum of routing variables $r_{ij}^k(t)$ must satisfy $\sum_{k\in\mathcal{K}} r_{ij}^k \leq C_{ij}$ the maximum objective is attained by making $r_{ij}^k(t) = C_{ij}$ for $k = k_{ij}^*$ and $r_{ij}^k(t) = 0$ for all $k \neq k_{ij}^*$.

The algorithm that follows from the solution of this maximization is summarized in Algorithm 2. The dual stochastic subgradient descent is implemented using node level protocols. At each time instance nodes send their multipliers $\lambda_i^k(t)$ to their neighbors. After receiving multiplier information from its neighbors, each node can compute the multiplier differentials $\lambda_i^k(t) - \lambda_j^k(t)$ for each edge. The node then allocates the full capacity of each outgoing link to whichever commodity has the largest differential. Once the transmission rates $r_{ij}^k(t)$ are set they are used for communication. Nodes also share these variables as shown in Step 7 so that they can be used to compute the stochastic gradient in (4.11). This stochastic gradients are used to update the multipliers as shown in Step 8 [cf. (4.13)].

Comparing (4.4) with (4.14) we see that the assignments of flows to links are the same if we identify multipliers $\lambda_i^k(t)$ with queue lengths. Furthermore, if we consider the Lagrangian update in (4.13) with $\alpha_t = 1$ for all times t and $\xi = 0$ we see that they too coincide. Thus, we can think of Backpressure as a particular case of stochastic subgradient descent. From that point of view algorithms 1 and 2 are identical. The only difference is that since queues take the place of multipliers the update in Step 8 of Algorithm 2 is not necessary in Algorithm 1. The exchange of routing variables $\{r_{ij}^k(t)\}_{kj}$ in Step 7 that is necessary to implement Step 8 is not needed as well.

4.1.3 Soft Backpressure

Assume that the functions $f_{ij}^k(r_{ij}^k)$ are continuously differentiable, strongly convcave, and monotone decreasing on $\mathbb{R}_+$. In this case the derivatives $y = \partial f_{ij}^k(x)/\partial x \leq 0$ for all $x \geq 0$ and thus have

Algorithm 3: Soft Backpressure at node i

```
1  Observe $q_i^k(0)$. Initialize $\lambda_i^k(0) = q_i^k(0)$ for all $k$ and $i \neq o_k$
2  for $t = 0, 1, 2, \cdots$ do
3      for all neighbors $j \in n(i)$ do
4          Send duals $\{\lambda_i^k(t)\}_k$ – Receive duals $\{\lambda_j^k(t)\}_k$
5          Compute $\mu_{ij}$ such that: $\sum_k F_{ij}^k \left( - \left[ \lambda_i^k - \lambda_j^k - \mu_{ij}(\lambda) \right]^+ \right) = C_{ij}$
6          Transmit packets at rate: $r_{ij}^k(t) = F_{ij}^k \left( -[\lambda_i^k(t) - \lambda_j^k(t) - \mu_{ij}]^+ \right)$
7      end
8      Send variables $\{r_{ij}^k(t)\}_{kj}$ – Receive variables $\{r_{ji}^k(t)\}_{kj}$
9      Update multipliers $\{\lambda_i^k(t)\}_k$ along stochastic subgradient
           $$\lambda_i^k(t+1) = \left[ \lambda_i^k(t) - \alpha_t \Big( \sum_{j \in n_i} r_{ij}^k(t) - r_{ji}^k(t) - a_i^k(t) - \xi \Big) \right]^+$$
10 end
```

inverse functions that we denote as

$$F_{ij}^k(y) := \left[\partial f_{ij}^k(x) / \partial x \right]^{-1} (y) \geq 0, \, \forall y \leq 0. \tag{4.15}$$

The Lagrangian maximizers in (4.10) can be explicitly written in terms of the derivative inverses $F_{ij}^k(y)$. Furthermore, the maximizers are unique for all λ implying that the dual function is differentiable. We detail these two statements in the following proposition.

Proposition 4. If the functions $f_{ij}^k(r_{ij}^k)$ in (4.5) are continuously differentiable, strongly convcave, and monotone decreasing on $\mathbb{R}_+$, the dual function $h(\lambda) := \max_{\sum_k r_{ij}^k \leq C_{ij}} \mathcal{L}(r, \lambda)$ is differentiable for all λ. Furthermore, the gradient component along the λ_i^k direction is $\bar{g}_i^k(\lambda)$ as defined in (4.11) with

$$R_{ij}^k(\lambda) = F_{ij}^k \left(- \left[\lambda_i^k - \lambda_j^k - \mu_{ij}(\lambda) \right]^+ \right). \tag{4.16}$$

where $\mu_{ij}(\lambda)$ is either 0 if $\sum_k F_{ij}^k \big(- \left[\lambda_i^k - \lambda_j^k \right]^+ \big) \leq C_{ij}$ or chosen as the solution to the equation

$$\sum_k F_{ij}^k \left(- \left[\lambda_i^k - \lambda_j^k - \mu_{ij}(\lambda) \right]^+ \right) = C_{ij}. \tag{4.17}$$

Proof : The dual problem (4.7) is convex because it is the Lagrange dual of a concave maximization problem. Since $f(r)$ is strongly concave, the maximizers $R(\lambda)$ generated by (4.10) are unique. Thus $\bar{g}(\lambda)$ is unique for any λ, ensuring that $h(\lambda)$ is differentiable, appendix A.4 [Boyd and Vandenberghe, 2004].

To compute the maximizers $R(\lambda)$, we consider the primal optimization (4.10). Taking the Lagrange dual of the domain constraints yields the extended Lagrangian,

$$\bar{\mathcal{L}}(r,\mu) = \sum_k f_{ij}^k(r_{ij}^k) + r_{ij}^k(\lambda_i^k - \lambda_j^k) + \mu_{ij}(C_{ij} - \sum_k r_{ij}^k). \tag{4.18}$$

Considering the Karush-Kuhn-Tucker (KKT) optimality conditions as defined in [Boyd and Vandenberghe, 2004][Section 5.5] for (4.18) yields the equations

$$f'_{ij,k}(r_{ij}^k) \quad = \quad -\left[\lambda_i^k - \lambda_j^k - \mu_{ij}\right]^+ \tag{4.19}$$

$$\sum_k r_{ij}^k \quad \leq \quad C_{ij} \tag{4.20}$$

for all $(i,j) \in \mathcal{E}$ where $f'_{ij,k}(x) = \partial f_{ij}^k(x)/\partial x$. Applying the definition of $F_{ij}^k(\cdot)$ from equation (4.15) to (4.19) we get the desired relation in (4.16). It remains to enforce (4.20) by selection of μ_{ij} which gives us condition (4.17). The assertion that

$$\mu_{ij} = 0 \text{ when } \sum_k F_{ij}^k\big(-\left[\lambda_i^k - \lambda_j^k\right]^+\big) \leq C_{ij} \tag{4.21}$$

holds by complementary slackness, Section 5.5 in [Boyd and Vandenberghe, 2004]. ■

While (4.17) does not have a closed form solution it can be computed quickly numerically using a binary search because it is a simple single variable root finding problem. We operate under the assumption that computation time cost is small compared to communication time cost.

The result in Proposition 4 combined with the stochastic subgradient descent iteration in (4.13) yields the Soft Backpressure algorithm summarized in Algorithm 3. Soft Backpressure is implemented using node level protocols. At each time instance nodes send their multipliers $\lambda_i^k(t)$ to their

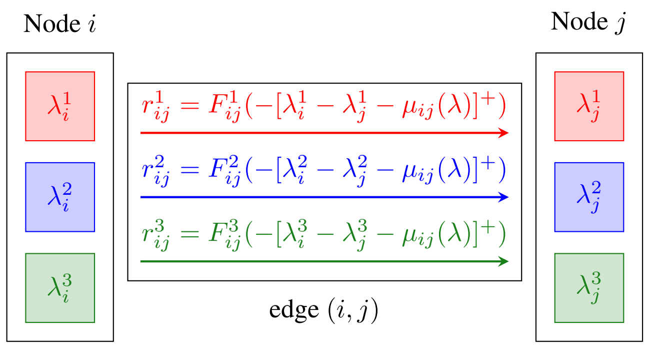

Node i
Node j
λ_i^1
λ_i^2
λ_i^3
$r_{ij}^1 = F_{ij}^1(-[\lambda_i^1 - \lambda_j^1 - \mu_{ij}(\lambda)]^+)$
$r_{ij}^2 = F_{ij}^2(-[\lambda_i^2 - \lambda_j^2 - \mu_{ij}(\lambda)]^+)$
$r_{ij}^3 = F_{ij}^3(-[\lambda_i^3 - \lambda_j^3 - \mu_{ij}(\lambda)]^+)$
edge (i, j)
λ_j^1
λ_j^2
λ_j^3

Figure 4.1: Flow variables r_{ij}^k encoding packet transmission rates are computed from dual variables at nodes i and j. For Backpressure and Soft Backpressure the variables λ_i^k are equivalent to the queue lengths q_i^k. For Accelerated Backpressure, λ_i^k is a priority rating for q_i^k. The constant $\mu_{ij}(\lambda)$ is chosen so that the total rate $\sum_k r_{ij}^k$ stays below the edge's capacity C_{ij}.

neighbors. After receiving multiplier information from its neighbors, each node can compute the multiplier differentials $\lambda_i^k(t) - \lambda_j^k(t)$ for each edge. The nodes then solve for μ_{ij} on each of its outgoing edges to satisfy (4.17). The capacity of each edge is then allocated to the different commodities using the expression in (4.16). With the transmission rates $\{r_{ij}^k(t)\}_{kj}$ set, we proceed as in regular Backpressure. The communication rates are used for communication and also shared with neighbors as shown in Step 7. These variables determine the stochastic gradient in (4.11) which is used to update the multipliers as shown in Step 8 [cf. (4.13)].

Algorithm 3 is called Soft Backpressure because rather than allocating all of an edge's capacity to the data type with the largest pressure $\lambda_i^k - \lambda_j^k$, it divides the capacity among the different data types based on their respective pressures via (4.16). This expression has a reverse water filling interpretation; see [Ribeiro, 2009b]. We further observe that Soft Backpressure is solved using the same information required of Backpressure, the lengths of the queues for all data types on either end of the link whose flow you are computing. This means that assuming a step size $\alpha_t = 1$ and constant $\xi = 0$, Algorithm 3 can be implemented without computing the multipliers but rather by simply observing the queue lengths $q_i^k(t)$ which are equal to the dual variables $\lambda_i^k(t)$.

An important difference between Backpressure and Soft Backpressure is that the former is equivalent to stochastic *subgradient* descent whereas the latter is tantamount to stochastic *gradi-*

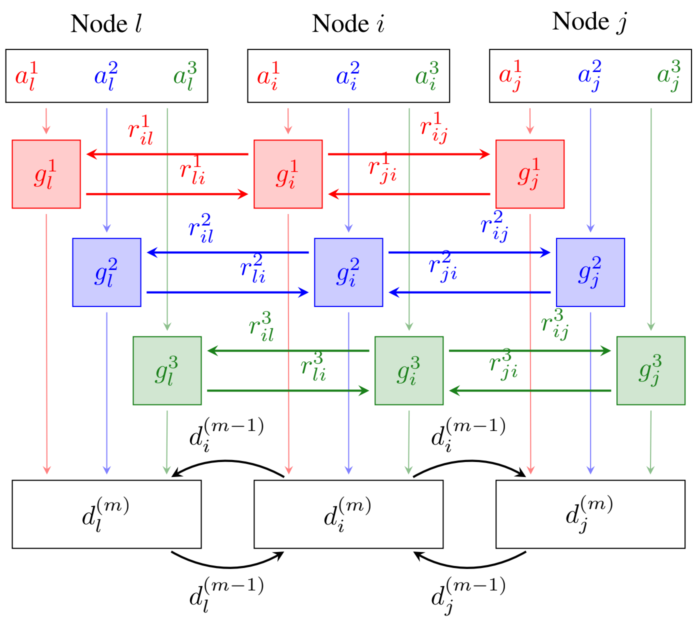


Figure 4.2: The Update of the queue priorities λ_i^k is determined by the value of the gradient component $g_i^k = \sum_{j\in n_i} r_{ji}^k - r_{ij}^k - a_i^k - \xi$ which is determined by the values of local and neighboring routing variables as well as the arrival rates observed from upper layers. In Backpressure and Soft Backpressure the gradient is just subtracted from λ_i^k. In Accelerated Backpressure the gradient is used along with local curvature information to determine the approximate Newton direction $d_i^{(N)}$. This is accomplished by the embedded communication loop that we summarize in Algorithm 5.

ent descent because the dual function is differentiable. This improves the average convergence rate from $O(1/\sqrt{k})$ to $O(1/k)$. In practice, Soft Backpressure is still slow to converge, motivating the ABP algorithm that we introduce in the following section.

4.2 Accelerated Backpressure

The backpressure-type algorithms of Section 4.1 exhibit slow convergence because they use rate of change information, but have no knowledge of curvature. In twice differentiable centralized deterministic problems, second order methods such as Newton method are used to take advantage of curvature information. Direct application of Newton-type methods is not possible here because: (i) the dual function is not twice differentiable; (ii) only a stochastic gradient can be observed in accordance with (4.11); (iii) the Newton direction requires data gathering and processing at a central node. We solve (i) by introducing a generalized Hessian; (ii) is resolved by the fact that the generalized Hessian is deterministic in this case and (iii) is resolved by using a Taylor's expansion of the Hessian inverse to approximate the Newton direction.

4.2.1 The Generalized Hessian

The dual function is not twice differentiable because the Lagrange dual defined in (4.7) has domain constraints which cause lack of smoothness in its gradient – see propositions 5 and 6. However, the second order behavior can be characterized by a *Generalized Hessian*, which is analagous to a subgradient because it may be taken from a set of values defined by the convex hull of the left and right limits approaching the points of discontinuity.

Definition 4.1 (Generalized Hessian [Sun and Kuo, 1998]). We say that $H(\lambda)$ is a generalized Hessian of the dual function $h(\lambda) = \mathcal{L}(R(\lambda), \lambda)$ if and only if

$$H(\lambda) \in \partial^2 \mathcal{L}(R(\lambda), \lambda) = co\left\{ \lim_{\bar{\lambda} \to \lambda, \bar{\lambda} \in D_g} \nabla^2 \mathcal{L}(R(\lambda), \lambda) \right\} \tag{4.22}$$

where co$\{\cdot\}$ denotes convex hull and D_g is the set of points on which $\mathcal{L}(R(\lambda), \lambda)$ is twice differentiable.

Writing $H(\lambda)$ in block form we have,

$$H(\lambda) = \begin{bmatrix} H_{11}(\lambda) & H_{12}(\lambda) & \cdots & H_{1n}(\lambda) \\ H_{21}(\lambda) & H_{22}(\lambda) & \dots & H_{2n}(\lambda) \\ \vdots & \vdots & \ddots & \vdots \\ H_{n1}(\lambda) & H_{n2}(\lambda) & \cdots & H_{nn}(\lambda) \end{bmatrix}. \tag{4.23}$$

The matrix (4.23) has $n \times n$ blocks where each of the blocks is a square matrix with dimension K

equal to the number of commodity types. The elements of the $(i,j)^{th}$ block are explicitly given by

$$\Big[H_{ij}(\lambda)\Big]_{kl} := \frac{\partial^2 h(\lambda)}{\partial\lambda_i^k \partial\lambda_j^l} = \frac{\partial^2 \mathcal{L}(R(\lambda),\lambda)}{\partial\lambda_i^k \partial\lambda_j^l}, \tag{4.24}$$

where we expressed the dual function $h(\lambda) = \mathcal{L}(R(\lambda),\lambda)$ in terms of the Lagrangian maximizers $R(\lambda)$ in (4.8) to write the second equality. As per (4.24), the $(i,j)^{th}$ block H_{ij} is associated with second derivatives with respect to dual variables of nodes i and j. The $(k,s)^{th}$ element of each block is associated with second derivatives with respect to commodities k and s.

Implicit in (4.24) is the differentiation of $R(\lambda)$ with respect to λ. It can be seen from (4.16) that $R_{ij}^k(\lambda)$ is not differentiable at the points where $\mu_{ij}(\lambda) = \lambda_i^k - \lambda_j^k$. These points arise precisely when the capacity constraint on edge (i,j) becomes active.

We show that D_g is a dense set by considering differentiability of $R_{ij}^k(\lambda)$ in two cases. First, consider the case with λ such that $\sum_k R_{ij}^k(\lambda) = C_{ij}$, then according to Proposition 4, $R_{ij}^k(\lambda)$ is may not be differentiable when $\lambda_i^k - \lambda_j^k = \mu_{ij}(\lambda)$, due to the $[\cdot]^+$ function. According to (4.15), when $\lambda_i^k - \lambda_j^k = \mu_{ij}(\lambda)$, we have $R_{ij}^k(\lambda) = F_{ij}^k(0) = \max_x f_{ij}^k(x)$. By monotonicity (decreasing) of $f_{ij}^k(x)$ on $\mathbb{R}^+$, the optimal routing is the function $R_{ij}^k(\lambda) = 0$ which is differentiable with $\nabla R_{ij}^k(\lambda) = 0$ for any λ such that $\lambda_i^k - \lambda_j^k < \mu_{ij}(\lambda)$. Therefore, given $\lambda_i^k - \lambda_j^k = \mu_{ij}(\lambda)$ when can choose $\bar{\lambda}$ with $\bar{\lambda}_i^k = \lambda_i^k - \epsilon$ for sufficiently small $\epsilon > 0$ and $\bar{\lambda}_j^k = \lambda_j^k$ for all $j \neq i$ ensuring $R_{ij}^k(\bar{\lambda})$ is differentiable.

Second, assume we have λ such that $\sum_k R_{ij}^k(\lambda) < C_{ij}$. Then according to Proposition 4, $\mu_{ij} = 0$ and $R_{ij}^k(\lambda)$ is differentiable as long as $\lambda_i^k \neq \lambda_j^k$. If $\lambda_i = \lambda_j$, there is a $\bar{\lambda}$ with $\bar{\lambda}_i^k = \lambda_i^k + \epsilon$ and $\bar{\lambda}_j = \lambda_j$ for all $i \neq j$ which is differentiable for ϵ small enough to ensure $\sum_k R_{ij}^k(\bar{\lambda}) < C_{ij}$; by continuity of $F_{ij}^k(x)$ a sufficiently small ϵ exists. Furthermore, when $\mu_{ij} = 0$ and $\lambda_i^k < \lambda_j^k$, we have $\lambda_i^k - \lambda_j^k < \mu_{ij}(\lambda)$ and $R_{ij}^k(\lambda) = 0$ which yields the property that the discontinuities all occur between non-trivial functions of λ and the zero function.

From Proposition 4, if the capacity constraint is not active, the gradient $\nabla_\lambda R_{ij}^k(\lambda)$ is a non-trivial function of λ, but when the capacity constraints becomes active $\nabla_\lambda R_{ij}^k(\lambda)$ is zero. Since the points of discontinuity are transitions between the nonzero gradients and zero gradients $\nabla_\lambda R_{ij}^k(\lambda) = 0$ the definition of the set of Generalized Hessians in (4.22) allows us to select the null derivative at these points to generate a specific Generalized Hessian.

We proceed to compute this specific Generalized Hessian. The set of commodities that are active across a particular edge are important. Thus, for given λ and edge $(i,j) \in \mathcal{E}$ define

$$\mathcal{K}_{ij}(\lambda) = \{k \in \mathcal{K} : R_{ij}^k(\lambda) > 0\}, \tag{4.25}$$

as the set of commodities that are being transported across given edge (i,j) for multipliers λ. Further define the capacity sharing coefficients

$$s_{ij}(\lambda) = \begin{cases} 1/|\mathcal{K}_{ij}(\lambda)| & \text{if } \mu_{ij}(\lambda) > 0, \\ 0 & \text{if } \mu_{ij}(\lambda) = 0, \end{cases} \tag{4.26}$$

which take the value zero when the capacity constraint on edge (i,j) is inactive, i.e., when $\sum_k r_{ij}^k(\lambda) < C_{ij}$, and equals the inverse cardinality $1/|\mathcal{K}_{ij}(\lambda)|$ of the set $\mathcal{K}_{ij}(\lambda)$ when the link capacity is saturated. Using these definitions computing the generalized Hessian $H(\lambda)$ is a straightforward but cumbersome derivation that we relegate to Appendix A. The results are stated in the following two propositions that concern the off-diagonal blocks $H_{ii}(\lambda)$ and diagonal blocks $H_{ii}(\lambda)$, respectively.

Proposition 5. Let $H_{ij}(\lambda)$ denote an off diagonal block of the generalized Hessian $H(\lambda)$ in (4.23) associated with the dual function $h(\lambda)$ defined in (4.7) and denote the derivative of the inverse derivative function $F_{ij}^k(\lambda)$ of (4.15) as $F'_{ij,k} = \partial F_{ij}^k(\lambda)/\partial\lambda$. The kth diagonal element of the Hessian block $H_{ij}(\lambda)$ is given by

$$\begin{aligned} \Big[H_{ij}(\lambda)\Big]_{kk} &= F'_{ij,k}\Big[R_{ij}^k(\lambda)\Big]\mathbf{1}\Big[k \in \mathcal{K}_{ij}(\lambda)\Big]\Big[1 - s_{ij}(\lambda)\Big] \\ &\quad + F'_{ji,k}\Big[R_{ji}^k(\lambda)\Big]\mathbf{1}\Big[k \in \mathcal{K}_{ji}(\lambda)\Big]\Big[1 - s_{ji}(\lambda)\Big]. \end{aligned} \tag{4.27}$$

where $\mathcal{K}_{ij}(\lambda)$ is the set of active commodities in the link $i \to j$ defined in (4.25) and $s_{ij}(\lambda)$ is the capacity sharing coefficient in (4.26). The (k,l)th off diagonal element of $H_{ij}(\lambda)$ can be written as

$$\begin{aligned}\Big[H_{ij}(\lambda)\Big]_{kl} &= -F'_{ij,k}\Big[R^k_{ij}(\lambda)\Big]\mathbf{1}\Big[k,l\in\mathcal{K}_{ij}(\lambda)\Big]s_{ij}(\lambda)\\ &\quad -F'_{ji,k}\Big[R^k_{ji}(\lambda)\Big]\mathbf{1}\Big[k,l\in\mathcal{K}_{ji}(\lambda)\Big]s_{ji}(\lambda).\end{aligned} \tag{4.28}$$

Proposition 6. The diagonal blocks $H_{ii}(\lambda)$ of the generalized Hessian $H(\lambda)$ in (4.23) associated with the dual function $h(\lambda)$ defined in (4.7) are given by the sums of the off diagonal blocks $H_{ij}(\lambda)$, i.e.,

$$H_{ii}(\lambda) \;=\; \sum_j H_{ij}(\lambda) \;=\; \sum_{j\in n_i} H_{ij}(\lambda). \tag{4.29}$$

4.2.2 Derivation of the Generalized Dual Hessian

Proof : We begin by computing the optimal flow rates $R^k_{ij}(\lambda)$ from the optimal queue priorities as defined in (4.16). Substituting into

$$\frac{\partial\mathcal{L}(R(\lambda),\lambda)}{\partial\lambda^k_i} = \sum_{j\in n_i} R^k_{ji}(\lambda) - R^k_{ij}(\lambda) - a^k_i \tag{4.30}$$

and differentiating with respect to λ we construct the Hessian. Since a^k_i is a constant the key is differentiating $R^k_{ij}(\lambda)$ with respect to λ. We can differentiate (4.16) using the chain rule yielding

$$\frac{\partial R^k_{ij}(\lambda)}{\partial y} = \frac{\partial F^k_{ij}(x)}{\partial x}\frac{\partial}{\partial y}\left(-\left[\lambda^k_i - \lambda^k_j - \mu_{ij}\right]^+\right) \tag{4.31}$$

where $y = \lambda^{k'}_{i'}$ for any $k', i' \neq o_{k'}$. The existence of $\partial F^k_{ij}(x)/\partial x$ is guaranteed by our assumptions on the edge costs f^k_{ij}. Differentiating $-\left[\lambda^k_i - \lambda^k_j - \mu_{ij}(\lambda)\right]^+$ is done by considering two cases. First, when the capacity constraint on edge (i,j) is inactive we have $\mu_{ij}(\lambda)=0$ from (4.21). In this case we have

$$\frac{\partial}{\partial\lambda^k_i}\left(-\left[\lambda^k_i - \lambda^k_j - \mu_{ij}(\lambda)\right]^+\right) = \begin{cases} -1 & \text{for } \lambda^k_i > \lambda^k_j \\ 0 & \text{for } \lambda^k_i \leq \lambda^k_j \end{cases} \tag{4.32}$$

and differentiating with respect to λ_j^k, we have

$$\frac{\partial}{\partial \lambda_j^k}\left(-\left[\lambda_i^k - \lambda_j^k - \mu_{ij}(\lambda)\right]^+\right) = \begin{cases} 1 & \text{for } \lambda_i^k > \lambda_j^k \\ 0 & \text{for } \lambda_i^k \leq \lambda_j^k \end{cases} . \tag{4.33}$$

There is no cross dependence between commodities k and l because $\mu_{ij} = 0$. Recall that a commodity is active on edge (i, j) if $\lambda_i^k > \lambda_j^k$, allowing us to substitute the condition $\lambda_i^k > \lambda_j^k$ for $k \in \mathcal{K}_{ij}$ in (4.32) and (4.33).

In the second case, the capacity constraint on edge (i, j) is active and $\mu_{ij}(\lambda) > 0$. By our design in Proposition 4, $\mu_{ij}(\lambda)$ produces the reverse water filling effect, producing

$$\frac{\partial}{\partial \lambda_i^k}\left(-\left[\lambda_i^k - \lambda_j^k - \mu_{ij}(\lambda)\right]^+\right) = \frac{1}{|\mathcal{K}_{ij}(\lambda)|} - 1 \tag{4.34}$$

and differentiating with respect to λ_j^k, we have

$$\frac{\partial}{\partial \lambda_i^k}\left(-\left[\lambda_i^k - \lambda_j^k - \mu_{ij}(\lambda)\right]^+\right) = 1 - \frac{1}{|\mathcal{K}_{ij}(\lambda)|} \tag{4.35}$$

where $\mathcal{K}_{ij}$ is the set of active commodities on the edge (i, j). Unlike in the previous case, there are cross terms:

$$\frac{\partial}{\partial \lambda_i^l}\left(-\left[\lambda_i^k - \lambda_j^k - \mu_{ij}(\lambda)\right]^+\right) = \frac{-1}{|\mathcal{K}_{ij}(\lambda)|}, \tag{4.36}$$

differentiating with respect to λ_j^l, we have

$$\frac{\partial}{\partial \lambda_i^l}\left(-\left[\lambda_i^k - \lambda_j^k - \mu_{ij}(\lambda)\right]^+\right) = \frac{1}{|\mathcal{K}_{ij}(\lambda)|}. \tag{4.37}$$

Discontinuities arise precisely when we transition from case one to case two which corresponds to the capacity constraint becoming active, i.e. an edge becoming saturated. In accordance with the generalized Hessian definition in (4.22), we can select any value in the convex hull of the limit points of our discontinuity. We proceed by selecting values produced by case 1, because many of the terms are null, greatly simplifying the computation. In fact case 1, also contains discontinuities when a commodity becomes active on an edge. In (4.32), our generalized Hessian definition allows us to choose

$$\frac{\partial}{\partial \lambda_i^k}\left(-\left[\lambda_i^k - \lambda_j^k - \mu_{ij}(\lambda)\right]^+\right) = 0 \tag{4.38}$$

when $\lambda_i^k = \lambda_j^k$ and the same holds for (4.33). To complete the proof we must place all the partial derivatives in their appropriate blocks. Observe that the blocks of $H(\lambda)$ correspond to node pairs (i, j). For pairs $(i, j) \notin \mathcal{E}$ the $H_{ij} = 0$ because there are no terms $R_{ij}^k(\lambda)$ for these pairs. Consider H_{ij} for $(i, j) \in \mathcal{E}$, generated by

$$[H_{ij}]_{kl} = \frac{\partial}{\lambda_i^k} g_j^l(\lambda) = \frac{\partial}{\lambda_i^k}\left(R_{ij}^l(\lambda) - R_{ji}^l(\lambda)\right) \tag{4.39}$$

which also holds in the case where $l = k$. For the diagonal blocks more of our partial derivative terms appear,

$$[H_{ii}]_{kl} = \frac{\partial}{\lambda_i^k} g_i^l(\lambda) = \sum_{j \in n_i} \frac{\partial}{\lambda_i^k}\left(R_{ij}^l(\lambda) - R_{ji}^l(\lambda)\right) \tag{4.40}$$

which also holds in the case where $l = k$. Applying the definitions of $\mathcal{K}_{ij}(\lambda)$ from (4.25), $s_{ij}(\lambda)$ from (4.26) and the partial derivatives from (4.32)-(4.37), completes the construction of the generalized Hessian. ■

4.2.3 Interpretation of the Generalized Dual Hessian

The expression in (4.29) reduces computation of the diagonal blocks of $H(\lambda)$ to the computation of the off diagonal blocks. Their validity endows the Hessian of the dual function with an interpretation as a weighted block Laplacian. The expressions in (4.27) and (4.28) look complicated but are actually simple. The terms of the form $F'_{ij,k}(R_{ij}^k(\lambda))$ are just the derivatives of the inverse derivative function in (4.15). If, e.g., we use quadratic functions $f_{ij}^k(R_{ij}^k(\lambda)) = -(R_{ij}^k(\lambda))^2/2$ we just have $F'_{ij,k}(R_{ij}^k(\lambda)) = -1$.

The indicator function $\mathbf{1}(k \in \mathcal{K}_{ij}(\lambda))$ in (4.27) simply distinguishes between the values of

λ for which commodity k is transported from i to j from those for which it is not. The term $F'_{ij,k}(R^k_{ij}(\lambda))(1 - s_{ij}(\lambda))$ is added to the Hessian component $[H_{ij}]_{kk}$ if and only if the commodity k is active in the link $i \to j$. Likewise the term $F'_{ji,k}(R^k_{ji}(\lambda))(1 - s_{ji}(\lambda))$ is added to Hessian component $[H_{ij}]_{kk}$ if and only if the commodity k is active in the link $j \to i$. The variables s_{ij} simply distinguish the case when $\mu_{ij}(\lambda)$ is null from when it is not, or, equivalently, the case when some of the link capacity is left unused from the case when all of the link capacity is allocated to some node. If some capacity is unused in the link $i \to j$ we have $\mu_{ij}(\lambda) = 0$ and the derivative $F'_{ij,k}(R^k_{ij})$ is not scaled. If all of the capacity in the link $i \to j$ is used, the derivative $F'_{ij,k}(R^k_{ij})$ is scaled by $(1 - 1/|\mathcal{K}_{ij}(\lambda)|)$.

Likewise, the indicator functions $\mathbf{1}(k, l \in \mathcal{K}_{ij}(\lambda))$ and $\mathbf{1}(k, l \in \mathcal{K}_{ji}(\lambda))$ in (4.28) imply that the corresponding terms are added to the Hessian component $[H_{ij}]_{kl}$ only if both commodities, k and l are active in the link $i \to j$ or the link $j \to i$. If some capacity is left unused in the link $i \to j$ we have $s_{ij}(\lambda) = 0$ and nothing is added to the Hessian component $[H_{ij}]_{kl}$. If all of the capacity is used we scale the derivative $F'_{ij,k}(R^k_{ij}(\lambda))$ by the inverse of the number of active commodities on the link, $1/|\mathcal{K}_{ij}(\lambda)|$. Likewise, we add nothing to $[H_{ij}]_{kl}$ if some capacity is left unused in the link $j \to i$ and scale $F'_{ji,k}(R^k_{ji}(\lambda))$ by the inverse of the number of active commodities on the link $j \to i$ otherwise.

Propositions 5 and 6 define a generalized Hessian for the dual function $h(\lambda)$ defined in (4.7). The components of the Hessian depend on the primal Lagrangian maximizers $R^k_{ij}(\lambda)$ as well as the auxiliary variable $\mu_{ij}(\lambda)$. As it follows from Proposition 4, both of these quantities depend on the dual variable λ but are independent of the arrival rates a^k_i. This is because the gradient dependence on the arrival rates is an additive constant [cf. (4.9)]. This simple fact is of fundamental importance here because it allows exact computation of the generalized Hessian $H(\lambda)$ and the consequent implementation of a stochastic Newton algorithm in which generalized Hessian inverses $H^{-1}(\lambda)$ premultiply stochastic gradients $g^k_i(\lambda)$ instead of (regular) gradients $\bar{g}^k_i(\lambda)$. Indeed, consider time index t and current iterate $\lambda(t)$ and recall that $g(t)$ stands for the stochastic subgradient with components given by (4.11). The stochastic Newton method replaces the dual update in (4.12) by the recursion

$$\lambda(t+1) = \Big[\lambda(t) - \alpha_t H^{-1}(t) g(t)\Big]^+, \tag{4.41}$$

where we have used $H^{-1}(t) = H^{-1}(\lambda(t))$ to denote the inverse of the generalized Hessian evaluated at $\lambda(t)$. The deterministic nature of the generalized Hessian $H^{-1}(t)$ when $\lambda(t)$ is given permits the definition of the dual stochastic Newton algorithm in (4.41). This possibility notwithstanding, implementation of the stochastic Newton method in (4.41) requires centralized computation of the generalized Hessian inverses $H^{-1}(t)$. The remaining challenge addressed by ABP is to approximate the inversion of $H(\lambda)$ through local operations. We discuss this in the following section.

4.2.4 Distributed Approximation of the Stochastic Newton Step

We begin by pointing out that the generalized Hessian in (4.23) with blocks as explicitly given in propositions 5 and 6 is block sparse with a sparsity pattern that matches the adjacency matrix of the network. We state this in the following lemma along with the fact that the diagonal blocks $H_{ii}(\lambda)$ are positive semidefinite.

Lemma 8. The Dual Hessian, $H(\lambda)$ in (4.23) associated with the dual function $h(\lambda)$ defined in (4.7) is block sparse with respect to the graph $\mathcal{G}$, i.e.,

$$H_{ij}(\lambda) = \mathbf{0} \qquad \text{for all } i \neq j, \text{ s.t. } (i,j) \notin \mathcal{E}. \tag{4.42}$$

Furthermore, the diagonal blocks of $H_{ii}(\lambda)$ are positive semidefinite.

Proof : From (4.26), $\sum_k \mathbf{1}(k \in \mathcal{K}_{ij}) s_{ij} = 1$, combined with (4.29), we have $[H_{ii}]_{kk} \geq \sum_l [H_{ii}]_{kl}$. Recall, $F'_{ij,k}(r_{ij}^k) \leq 0$ from (4.15) and the assumption that $f_{ij}^k(\cdot)$ is monotone decreasing. Therefore the diagonal elements are positive and according to Section 6.2 of [Berman and Plemmons, 1979], H_{ii} is positive semidefinite by diagonal dominance. Block sparsity arises trivially from the fact that $\partial/\partial\lambda_{i'}^{k'} \left(-\Big[\lambda_i^k - \lambda_j^k - \mu_{ij}\Big]^+\right) = 0$ for any k' when i' is not i or j. ■

Lemma 8 guarantees that all elements of the Hessian can be computed using local information. Elements of the Hessian require knowledge of the local action sets $\mathcal{K}_{ij}(\lambda)$ from (4.25), which are computed using the local flow values $\{r_{ij}^k(\lambda)\}_k$. Furthermore, Lemma 8 states that the diagonal blocks $H_{ii}(\lambda)$ are positive semidefinite, which in turn indicates that the nodes depend positively on their own queues. This fits our intuition because we expect penalties on a specific queue to become larger when those queues grow.

In order to accelerate Backpressure and retain its distributed nature we compute a dual update direction based on the ADD family of algorithms defined in [Zargham et al., 2011]. ADD relies on splitting the dual Hessian and leverages its block sparsity pattern to approximate its inverse. To be explicit, consider the dual iterate $\lambda(t)$ at time t and denote as $H(t) = H(\lambda(t))$ the corresponding Hessian. Let I denote an identity matrix of proper dimension and define the block diagonal matrix $D(t)$ with diagonal blocks $D_{ii}(t)$ defined in terms of the Hessian diagonal blocks $H_{ii}(t)$ as

$$D_{ii}(t) = 2H_{ii}(t) + I. \tag{4.43}$$

The Hessian splitting separates the block diagonal matrix $D(t)$ for which we define a matrix $B(t) := D(t) - H(t)$ so that we can write

$$\begin{aligned} H(t) &= D(t) - B(t) \\ &= D(t)\left[I - D^{-1}(t)B(t)\right]. \end{aligned} \tag{4.44}$$

For future reference observe that $B(t)$ has the same sparsity pattern of $H(t)$. Denote the diagonal blocs of $B(t)$ as $B_{ii}(t) = D_{ii}(t) - H_{ii}(t) = I + H_{ii}(t)$ and the off diagonal blocks of $B(t)$ as $B_{ij}(t) = -H_{ij}(t)$

The ADD-N algorithm approximates the Hessian inverse $H^{-1}(t)$ by truncating the Taylor's expansion of the term $\left[I - D^{-\frac{1}{2}}(t)B(t)D^{-\frac{1}{2}}(t)\right]^{-1}$ at its Nth term. Using $\bar{H}^{(N)}$ to denote such approximation we have

$$\bar{H}^{(N)}(t) = \sum_{m=0}^{N} \left[D^{-1}(t)B(t)\right]^m D^{-1}(t). \tag{4.45}$$

Relying on the Hessian inverse approximation in (4.45) to substitute $H^{-1}(t)$ in the stochastic Newton method in (4.41) yields the ABP-N algorithm. Specifically, define the ABP-N direction as

$$\begin{aligned} d^{(N)}(t) \; &:= \; -\bar{H}^{(N)}(t)g(t) \\ &= -\sum_{m=0}^{N} \left[D^{-1}(t)B(t)\right]^{m} D^{-1}(t)g(t), \end{aligned} \tag{4.46}$$

and the ABP-N algorithm by recursive application of the iteration

$$\lambda(t+1) = \left[\lambda(t) - \alpha_t \bar{H}^{(N)}(t)g(t)\right]^{+} = \left[\lambda(t) + \alpha_t d^{(N)}(t)\right]^{+}. \tag{4.47}$$

Equivalently, we can define the local components $d_i^{(N)}(t)$ of the ABP direction $d_i^{(N)}(t) = \{d_i^{(N)}(t)\}_{i=1}^{n}$ so that it is possible to rewrite (4.47) for each of the multiplier components,

$$\lambda_i(t+1) = \left[\lambda_i(t) + \alpha_t d_i^{(N)}(t)\right]^{+}. \tag{4.48}$$

The important observation here is that the ABP-N direction $d_i^{(N)}(t)$ in (4.48) can be computed by node i by aggregating information from, at most, N hops away. Indeed, observe that $B(t)$ has a sparsity pattern that matches the sparsity pattern of the network, because this is true of $H(t)$. Thus, when we consider the second summand $\left[D^{-1}(t)B(t)\right]^{1}$ in (4.46) we end up with a matrix whose sparsity pattern is that of the underlying network. When we consider the third term $\left[D^{-1}(t)B(t)\right]^{2}$ the resulting sparsity pattern is that of the network of two-hop neighbors. Since determination of the ABP direction $d^{(N)}(t)$ necessitates communication with the nodes that match this sparsity pattern, no information from nodes farther than N hops away is necessary.

More to the point, observe that the ABP-0 direction can be written as $d^{(0)} = -D^{-1}g_i$ and that for all N other than 0 we can write the recursion

$$d^{(N)}(t) = -D^{-1}(t)g(t) + D^{-1}(t)B(t)d^{(N-1)}(t). \tag{4.49}$$

Exploiting the fact that the sparsity pattern of $B(t)$ is that of the network and using the definitions of the local pieces $d_i^{(N)}$ of the ABP direction and $g_i(t)$ of the gradient as well as the definition of the blocks D_{ii} and B_{ij} the recursion in (4.49) is equivalent to the componentwise recursions

$$d_i^{(N)}(t) = D_{ii}^{-1}(t)g_i(t) + \sum_{j\in n_i, j=i} D_{ii}^{-1}(t)B_{ij}(t)d_j^{(N-1)}(t), \tag{4.50}$$

where the ABP-0 direction can also be computed locally as $d_i^{(0)}(t) = -D_{ii}^{-1}(t)g_i(t)$. The expres-

Algorithm 4: Local computation of ABP direction

```
1  function $d_i^N$ = ABP-N dir $\left(\{r_{ij}^k, r_{ji}^k\}_{jk}, \{a_i^k\}_k, \{\mu_{ij}\}_j\right)$
2  Active sets $\mathcal{K}_{ij}$ as per (4.25). Sharing coefficients $s_{ij}$ as per (4.26)
3  Hessian blocks $H_{ij}$ as per (4.27) and (4.28). Blocks $H_{ii}$ as per (4.29)
4  Set $D_{ii} = 2H_{ii} + I$, $B_{ii} = I + H_{ii}$ and $B_{ij} = -H_{ij}$
5  Stochastic gradients: $g_i^k = \sum_{j\in n_i} r_{ji}^k - r_{ij}^k - a_i^k - \xi$
6  Initialize ABP direction: $d_i^{(0)} = -D_{ii}^{-1} g_i$,
7  for $m$=1,..., $N$ do
8      Send $d_i^{(m-1)}$ to $j \in n_i$. Receive directions $d_j^{(m-1)}$ from $j \in n_i$
9      Update ABP direction: $d_i^{(m)} = -D_{ii}^{-1} g_i(t) + \sum_{j\in n_i} D_{ii}^{-1} B_{ij} d_j^{(m-1)}$
10 end
11 return $d_i^{(N)} = d_i^{(m)}$
```

sion in (4.50) results in a local function for the computation of the local ABP direction $d_i(t)$ that we shown in Algorithm 4. The inputs to this function are routing rates $r_{ij}^k = r_{ij}^k(t)$ and $r_{ji}^k = r_{ji}^k(t)$ for all neighbors $j \in n_i$ and flows k; arrival rates $a_i^k = a_i^k(t)$ for all flows k; and water levels μ_{ij} for all neighbors $j \in n_i$. All of these variables are available locally except for the incoming rates r_{ji}^k that are communicated from neighbors. These inputs are used to determine the active sets $\mathcal{K}_{ij}$ in (4.25) and the capacity sharing coefficients s_{ij} in (4.26) as stated in Step 2. Observe that in (4.25) and (4.26) these sets and coefficients are defined as functions of λ, which they are, but that is suffices to know r_{ij}^k, r_{ji}^k, and μ_{ij} to compute them. These preliminary computations are then used in Step 3 to determine the local Hessian blocks H_{ij} using the expressions (4.27) and (4.28) of Proposition 5 and the Hessian blocks H_{ii} using (4.29) in Proposition 6. The values of the Hessian block are then used to determine the corresponding blocks of the splitting matrices in (4.44). These blocks are $D_{ii} = 2H_{ii} + I$, $B_{ii} = I + H_{ii}$ and $B_{ij} = -H_{ij}$, which are computed in Step 4. The local components of the stochastic gradient g_i^k are then computed as dictated by (4.11). These preliminary computations lead to the core of the algorithm in steps 6-10 that implement the recursion in (4.50). Step 6 initializes the recursion by determining the value of the ABP-0 direction $d_i^{(0)} = -D_{ii}^{-1} g_i$. The update in Step 9 utilizes local and neighboring components of the ABP-$(m-1)$ direction to compute the local component of the ABP-m direction. To have the neighboring components available for this computation the local components of the ABP-$(m-1)$ direction have to be shared between neighbors as stated in Step 8. The outcome of N iterations of this loop is the ABP-N which is returned in Step 11. Observe that to implement this loop we require a total of N communication exchanges with neighboring nodes. ABP is summarized in Algorithm 5. The algorithm

Algorithm 5: Accelerated Backpressure for node i

```
1  Observe $q_i^k(0)$. Initialize $\lambda_i^k(0) = q_i^k(0)$ for all $k$ and $i \neq o_k$
2  for $t = 0, 1, 2, \cdots$ do
3      for all neighbors $j \in n(i)$ do
4          Send duals $\{\lambda_i^k(t)\}_k$ – Receive duals $\{\lambda_j^k(t)\}_k$
5          Compute $\mu_{ij}$ such that
               $$\sum_k F_{ij}^k \left( -\left[\lambda_i^k - \lambda_j^k - \mu_{ij}(\lambda)\right]^+ \right) = C_{ij}$$
6          Transmit packets at rate
               $$r_{ij}^k(t) = F_{ij}^k \left( -[\lambda_i^k(t) - \lambda_j^k(t) - \mu_{ij}]^+ \right)$$
7      end
8      Send variables $\{r_{ij}^k(t)\}_{kj}$ – Receive variables $\{r_{ji}^k(t)\}_{kj}$
9      $d_i^{(N)}(t)$ = ABP-N dir $\left( \{r_{ij}^k(t), r_{ji}^k(t)\}_{jk}, \{a_i^k(t)\}_k, \{\mu_{ij}\}_j \right)$
10     Update dual variables: $\lambda_i(t+1) = \left[\lambda_i(t) + \alpha_t d_i^{(N)}(t)\right]^+$
11 end
```

is identical to Soft Backpressure up until Step 8 and is repeated for clarity. The difference is that instead of updating the dual variables by descending on the local stochastic gradient we descend along the local ABP-N direction $d_i^{(N)}(t)$ (Step 10). The ABP-N direction is computed with a call to Algorithm 4 to which the routing rates $r_{ij}^k(t)$ and $r_{ji}^k(t)$, the arrival rates $a_i^k = a_i^k(t)$, and the water levels μ_{ij} are passed as parameters (Step 8).

Table 4.1: Maximum communication cost per node and iteration, relative to the maximum degree Δ and ABP parameter N.

	Communication	Computation
Backpressure	$O(\Delta)$	$O(\Delta)$
Soft Backpressure	$O(2\Delta)$	$O(3\Delta)$
Accelerated Backpressure	$O((N+2)\Delta)$	$O(N\Delta^2 + 3\Delta)$

4.2.5 Communication and Computationation Costs

Implementation of each ABP-N iteration requires $N+2$ communications with each neighbor. These include the exchange of duals and primals in steps 4 and 8 of Algorithm 5 and the N exchanges of ABP directions in Step 9 of Algorithm 4. This is an extra N communications per neighbor relative to the cost of Soft Backpressure – which requires only exchanges of primals and duals – and an extra $N + 1$ communications relative to Backpressure – which requires exchange of primals only. Denoting the maximum node degree as Δ this gives communication costs of order $O((N + 2)\Delta)$, $O(2\Delta)$, and $O(\Delta)$ as summarized in Table 4.1. The computational cost of each Backpressure iteration grows proportional to number of neighbors as we need to perform primal computations for each link. For Soft Backpressure we have primal and dual computations in steps 6 and 9 of Algorithm 3 each of which grows with the number of neighbors. We also need to compute the water level μ_{ij} as per Step 5 of Algorithm 3, the complexity of which also scales linearly with the number of neighbors. This yields a total complexity that scales in the order of 3Δ. For ABP-N we also have to implement N matrix products at a cost of not more than Δ^2 operations each. The resulting computational costs are also summarized in Table 4.1. We emphasize that all these costs grows with the maximum degree Δ rather than with the size of the network n. The communications and computation costs of ABP grow with the parameter N. Larger N generates a more accurate approximation of the Newton Step at the cost of additional communication and computation. In practice, implementations with $N = 1$ perform best in practice; see Section 4.4.

4.3 Stability Analysis

In order to claim the ABP algorithm is an alternative to the Backpressure and Soft Backpressure algorithms, a guarantee that it achieves queue stability is provided. We do so by combining a Lyapunov drift analysis [Georgiadis et al., 2006] with duality theory and supermartingale analysis, in particular as applied to stochastic subgradient descent, [Nedić and Ozdaglar, 2009b].

4.3.1 Preliminary Analysis

For compactness we make use of matrix vector notation and annotate time as subindices instead of arguments. The queue update equation in (4.1) becomes

$$q_{t+1} = q_t - g(\lambda_t) \tag{4.51}$$

where $q_t = \{q_i^k(t)\}$, and $g(\lambda_t) = \{g_i^k(\lambda_t)\}$ both of which are organized as $(n-1)$ stacks of $|\mathcal{K}|$-dimensional vectors. For the dual update in (4.47) we also remove the projection operator and write

$$\lambda_{t+1} = \lambda_t + \alpha_t d_t \tag{4.52}$$

where $\lambda_t = \lambda_(t)$ and $d_t = d^{(N)}(t)$. We can remove the projection in (4.47) because in problem (4.5) the capacity constraint is binding at the optimal point, making the problem equivalent to the equality constrained case. We proceed with our analysis using (4.52) and start by proving important properties of stochastic gradients and Hessians.

Lemma 9. The stochastic gradient is bounded

$$\|g_t(\lambda)\| \leq \gamma, \qquad \forall t, \forall \lambda, \tag{4.53}$$

every Generalized Hessian satisfies

$$\|H(\lambda)\| \leq \Gamma, \qquad \forall \lambda \tag{4.54}$$

and the approximate inverse Hessian is bounded

$$0 < \delta \leq \|\bar{H}(\lambda)\| \leq N + 1, \qquad \forall \lambda \tag{4.55}$$

where N is the communication parameter introduced in (4.45).

Proof : For $g_t(\lambda)$, consider (4.11) and recall our assumption that the random portion of the arrivals has finite support. Then observing from (4.8) that $R(\lambda)$ lies in a compact polyhedral set

$$R(\lambda) \in \mathcal{C} = \{r \geq 0 : \sum_k r_{ij}^k \leq C_{ij}, \, \forall (i,j) \in \mathcal{E}\}, \tag{4.56}$$

implies that $g_t(\lambda)$ has a finite upper bound γ that holds for all λ and all t.

In the case of $H(\lambda)$, we follow the same logic as for $g_t(\lambda)$. Consider any Generalized Hessian definition satisfying (4.22) and the element-wise definition in (4.24). $H(\lambda)$ is a function of $R(\lambda)$ because $\nabla h(\lambda) = \bar{g}(\lambda)$ defined in (4.9). From (4.25) and (4.26), $\mathcal{K}_{ij}(\lambda)$ and $s_{ij}(\lambda)$ are computed from $R(\lambda)$, so we can express

$$\|H(\lambda)\| = \|\tilde{H}(R(\lambda))\| \tag{4.57}$$

using Propositions 5 and 6. From the derivation of these propositions in Appendix A, $\tilde{H}(r)$ is finite, for finite r for any Generalized Hessian. Since $R(\lambda)$ lies in the compact polyhedral set from (4.56), we have

$$\|H(\lambda)\| \leq \max_{r \in \mathcal{C}} \|\tilde{H}(r)\| = \Gamma. \tag{4.58}$$

In the case of $\bar{H}_t(\lambda)$, we consider its construction for an arbitrary Generalized Hessian $H = H(\lambda)$. Observing equation (4.45) and considering the splitting $H = D - B$, each term in the sum (4.45) is positive definite with largest eigenvalue upper bounded by $\max_i \text{eig}\big((D_{ii})^{-1}\big)$. Since

$$D_{ii} = H_{ii} + I \succeq I$$

where $D_{ii} \succeq 0$ for any λ, we have $\|\bar{H}^{(N)}(\lambda)\| \leq N + 1$.

Since each term in the sum from (4.45) is positive, we consider the first term for which a lower bound δ is given by $\min_i \operatorname{eig}(D_{ii}^{-1}) = 1/(1 + \max_i \operatorname{eig}(H_{ii}))$ where $\max_i \operatorname{eig}(H_{ii})$ is precisely the maximum achievable eigenvalue of a diagonal block of $H(\lambda)$. Since each $H(\lambda)$ can be expressed as $\tilde{H}(R(\lambda))$ the maximum achievable value is finite for $R(\lambda) \in \mathcal{C}$, guaranteeing that our lower bound δ is strictly positive. ■

Lemma 10. The dual function $h(\cdot)$ satisfies,

$$h(\hat{\lambda}) - h(\lambda) \leq \bar{g}(\lambda)'(\lambda - \hat{\lambda}) + \frac{\Gamma}{2}\|\lambda - \hat{\lambda}\|^2 \tag{4.59}$$

where Γ is the upper bound for all generalized Hessians defined in Lemma 9.

Proof : We begin with the statement of the generalized second order Taylor expansion taken directly from item (iv) of section 2 in [Guerraggio et al., 2001]. Given a once continuously differentiable function $h(\lambda)$ with gradient $\bar{g}(\lambda)$ and set of generalized Hessians $H(\lambda)$,

$$h(\lambda) - h(\hat{\lambda}) - \bar{g}(\lambda)'(\hat{\lambda} - \lambda) \in \mathcal{S}(\lambda, \hat{\lambda}) \tag{4.60}$$

where the set $\mathcal{S}(\lambda, \hat{\lambda})$ is defined

$$\mathcal{S}(\lambda, \hat{\lambda}) = \left\{ \frac{1}{2}(\lambda - \hat{\lambda})' H (\lambda - \hat{\lambda}) : H \in \cup_{z \in [\lambda, \hat{\lambda}]} H(z) \right\}. \tag{4.61}$$

Consider the maximal element in $\bar{s}(\lambda, \hat{\lambda}) = \max \mathcal{S}(\lambda, \hat{\lambda})$,

$$\bar{s}(\lambda, \hat{\lambda}) = \max_{H \in \cup_{z \in [\lambda, \hat{\lambda}]} H(z)} \frac{1}{2}(\lambda - \hat{\lambda})' H (\lambda - \hat{\lambda}). \tag{4.62}$$

From the uniform upper bound on all generalized Hessians, $||H|| \leq \Gamma$, we have

$$\bar{s}(\lambda, \hat{\lambda}) \leq \frac{\Gamma}{2}(\lambda - \hat{\lambda})'(\lambda - \hat{\lambda}). \tag{4.63}$$

Since $h(\hat{\lambda}) - h(\lambda) - \bar{g}(\lambda)'(\hat{\lambda} - \lambda)$ belongs to the set $S(\lambda, \hat{\lambda})$, it can be no larger than the maximal element in that set,

$$h(\hat{\lambda}) - h(\lambda) - \bar{g}(\lambda)'(\hat{\lambda} - \lambda) \leq \bar{s}(\lambda, \hat{\lambda}). \tag{4.64}$$

Applying (4.63) to (4.64), completes the proof. ■

4.3.2 Supermartingale Analysis

Proposition 7. Consider the ABP algorithm implemented with the dual step (4.52), decaying step size α_t satisfying $\sum_t \alpha_t = \infty$, $\sum_t \alpha_t^2 < \infty$ and the ADD-N update direction $d_t^{(N)}$ as defined in (4.46), then

$$\lim_{t\to\infty} \|\bar{g}(\lambda_t)\| = 0 \tag{4.65}$$

almost surely.

Proof : Applying Lemma 10, for $\lambda = \lambda_t$ and $\hat{\lambda} = \lambda_{t+1}$, we have

$$h(\lambda_{t+1}) \leq h(\lambda_t) + \bar{g}(\lambda_t)'(\lambda_{t+1} - \lambda_t) + \frac{\Gamma}{2}\|\lambda_{t+1} - \lambda_t\|^2. \tag{4.66}$$

Define the dual optimality gap

$$L_t = h(\lambda_t) - h(\lambda^*). \tag{4.67}$$

Subtracting $h(\lambda^*)$ from both sides of (4.66), using the definition in (4.67), and observing that according to the dual ABP update in (4.52) we have $\lambda_{t+1} - \lambda_t = -\alpha_t \bar{H}(\lambda_t) g_t(\lambda_t)$ we can write

$$L_{t+1} \leq L_t - \alpha_t \bar{g}(\lambda_t)'\bar{H}(\lambda_t)g_t(\lambda_t) + \frac{\Gamma}{2}\|\alpha_t \bar{H}(\lambda_t)g_t(\lambda_t)\|^2. \tag{4.68}$$

Further recall the upper bound $\|\bar{H}(\lambda)\| \leq 2$ on the norm of the approximate Hessian inverse as well as the upper bound $\|g_t(\lambda)\| \leq \gamma$ on the norm of the stochastic gradient, both derived in Lemma 9. Using these bounds in the third summand on the right hand side we simplify (4.68) to

$$L_{t+1} \leq L_t - \alpha_t \bar{g}(\lambda_t)'\bar{H}(\lambda_t)g_t(\lambda_t) + \frac{\Gamma}{2}\alpha_t^2(N+1)^2\gamma^2. \tag{4.69}$$

Let $\lambda_{0:t}$ stand in for the history of the dual variables process up until time t and consider the expectation $\mathbb{E}\left[L_{t+1} \,\middle|\, \lambda_{0:t}\right]$ of the duality gap at time $t+1$ given this past history. With $\lambda_{0:t}$ given, λ_t in particular is given in (4.69). Since the duality gap L_t and the Hessian estimate $H(\lambda_t)$ depend on λ_t only, $H(\lambda_t)$ and L_t are also given in (4.69) and the only random variable left in the right hand side is $g_t(\lambda_t)$. Thus, it follows from (4.69) that the conditional expectation $\mathbb{E}\left[L_{t+1} \,\middle|\, \lambda_{0:t}\right]$ can be bounded as

$$\mathbb{E}\left[L_{t+1} \,\middle|\, \lambda_{0:t}\right] \leq L_t - \alpha_t \bar{g}(\lambda_t)' \bar{H}(\lambda_t) \mathbb{E}\left[g_t(\lambda_t) \,\middle|\, \lambda_{0:t}\right] + \frac{\Gamma}{2}\alpha_t^2 (N+1)^2 \gamma^2. \tag{4.70}$$

By definition, the stochastic gradient $g_t(\lambda_t)$ is such that $\mathbb{E}\left[g_t(\lambda_t) \,\middle|\, \lambda_{0:t}\right] = \mathbb{E}\left[g_t(\lambda_t) \,\middle|\, \lambda_t\right] = \bar{g}(\lambda_t)$. Substituting this equality in (4.70) yields

$$\mathbb{E}\left[L_{t+1} \,\middle|\, \lambda_{0:t}\right] \leq L_t - \alpha_t \bar{g}(\lambda_t)' \bar{H}(\lambda_t) \bar{g}(\lambda_t) + \frac{\Gamma}{2}\alpha_t^2 (N+1)^2 \gamma^2. \tag{4.71}$$

Observe that the quadratic form $\bar{g}(\lambda_t)' \bar{H}(\lambda_t) \bar{g}(\lambda_t)$ is positive definite since the smallest eigenvalue of $\bar{H}(\lambda)$ is bounded by a nonnegative constant δ as shown in Lemma 9. Using this eigenvalue lower bound further reduces (4.71) to

$$\mathbb{E}\left[L_{t+1} \,\middle|\, \mathcal{F}_t\right] \leq L_t - \alpha_t \delta \|\bar{g}(\lambda_t)\| + \frac{\Gamma}{2}\alpha_t^2 (N+1)^2 \gamma^2. \tag{4.72}$$

The sequences L_t, $\alpha_t \delta \|\bar{g}(\lambda_t)\|$, and $\frac{\Gamma}{2}\alpha_t^2 (N+1)^2 \gamma^2$ are nonnegative. The sequence $\frac{\Gamma}{2}\alpha_t^2 (N+1)^2 \gamma^2$ is also summable because square summability is a condition on the step size sequence α_t. These properties allow application of the supermartingale convergence theorem, see e.g., Theorem E.7.4 of [Solo and Kong, 1995]. Given nonnegative stochastic processes $V_t \geq 0$, $X_t \geq 0$ and $Y_t \geq 0$ and a sigma algebra $\mathcal{F}_t$ measuring the processes up until time t. If the processes are such that

$$\mathbb{E}\left[V_{t+1} \,\middle|\, \mathcal{F}_t\right] \leq V_t - X_t + Y_t, \tag{4.73}$$

and the process Y_t is summable with probability 1 then X_t is almost surely summable and that the limit of V_t exists almost surely, i.e.,

$$\lim_{t\to\infty} V_t = V_\infty, \quad \sum_{t=1}^{\infty} X_t < \infty, \quad \text{a.s.} \tag{4.74}$$

Identifying $\mathcal{F}_t$ with $\lambda_{0:t}$, V_t with L_t, X_t with $\alpha_t\delta\|\bar{g}(\lambda_t)\|$ and Y_t with $\Gamma/2\alpha_t^2(N+1)^2\gamma^2$ it follows from (4.74) that

$$\lim_{t\to\infty} L_t = L_\infty, \qquad \sum_{t=0}^{\infty} \alpha_t\delta\|\bar{g}(\lambda_t)\|^2 < \infty, \quad \text{a.s.} \tag{4.75}$$

We prove that L_∞ not only exists but is zero for almost all realizations. To do so, consider a realization for which $L_\infty \neq 0$. Then, the limit must be strictly positive because the sequence L_t is nonnegative. In turn, this implies that there exists a time τ such that for all $t \geq \tau$ we have $L_t \geq \kappa_1$ for some positive constant $\kappa_1 > 0$. Since $L_t = h(\lambda_t) - h(\lambda^*)$ this means that $h(\lambda_t) \geq h(\lambda^*) + \kappa_1$ for all times $t \geq \tau$. Recall now that for a differentiable convex function we can have $\nabla h(\lambda) = 0$ only when $\lambda = \lambda^*$; this is true even if $h(\lambda)$ is not differentiable, as long as we reinterpret $\nabla h(\lambda)$ as a subgradient. Using this fact it follows that if $h(\lambda_t) \geq h(\lambda^*) + \kappa_1$ for all $t \geq \tau$ there must be a constant $\kappa_2 > 0$ such that

$$\|\nabla h(\lambda_t)\| \geq \kappa_2 > 0, \quad \text{for all } t \geq \tau. \tag{4.76}$$

Observe now that by definition $\bar{g}(\lambda_t) = \nabla h(\lambda_t)$. Thus, we can use (4.76) to lower bound the series in (4.75) as

$$\sum_{t=0}^{\infty} \alpha_t\delta\|\bar{g}(\lambda_t)\|^2 \;\geq\; \sum_{t=\tau}^{\infty} \alpha_t\delta\|\bar{g}(\lambda_t)\|^2 \;\geq\; \kappa_2^2\delta\sum_{t=\tau}^{\infty} \alpha_t. \tag{4.77}$$

Since the step size sequence is nonsummable by assumption, we have that $\sum_{t=\tau}^{\infty} \alpha_t = \infty$ which, upon substitution in (4.77) yields

$$\sum_{t=0}^{\infty} \alpha_t\delta\|\bar{g}(\lambda_t)\|^2 = \infty \tag{4.78}$$

Since we know that (4.78) is true in, at most, a set of zero measure, and also true whenever $L_\infty \neq 0$, it must be that $L_\infty \neq 0$ in a set of zero probability. Thus, we must have $L_\infty = 0$ a.s. Further using the definition of $L_t = h(\lambda_t) - h(\lambda^*)$ we conclude

$$\lim_{t\to\infty} h(\lambda_t) = h(\lambda^*), \quad \text{a.s.} \tag{4.79}$$

As we already pointed out the dual function is differentiable according to Proposition 4. It then must satisfy the first order optimality condition $\bar{g}(\lambda^*) = \nabla h(\lambda^*) = 0$. Thus, the almost sure convergence of $h(\lambda_t)$ to $h(\lambda^*)$ stated in (4.79) implies almost sure convergence of $\bar{g}(\lambda^*) = \nabla h(\lambda^*)$ to 0 as stated in (4.65). ■

Proposition 7 is the first step in our theoretic convergence guarantee. By implementing a decaying step size when updating the dual variables, convergence to the optimal dual variables is achieved. Since the sequence of dual variables converges to the optimal dual variables almost surely, our routing $R^k_{ij}(\lambda_t)$ becomes a feasible routing. We proceed to leverage this fact to ensure the queues remain stable.

Proposition 8. Consider a dual variables process λ_t such that the dual gradient $\bar{g}(\lambda_t)$ satisfies

$$\lim_{t\to\infty} ||\bar{g}(\lambda_t)|| = 0, \quad \text{a.s.} \tag{4.80}$$

Then, all queues empty infinitely often with probability one, i.e.,

$$\liminf_{t\to\infty} q^k_i(t) = 0, \quad \text{a.s.,} \qquad \text{for all } k, i \neq o_k \tag{4.81}$$

Proof : The result in (4.81) is true if for arbitrary time $\tau \geq 0$ and arbitrary constant ϵ the queue length $q^k_i(t)$ is almost surely smaller than ϵ at some point the future. To write this statement formally define the stopped process

$$\tilde{q}^k_i(t) = \begin{cases} q^k_i(t) & \text{if } q^k_i(u) > 0 \text{ for all } \tau \leq u \leq t, \\ 0 & \text{else .} \end{cases} \tag{4.82}$$

The stopped process $\tilde{q}^k_i(t)$ follows the queue length process $q^k_i(t)$ until the queue length becomes null for the first time after time τ. If and when the queue empties after time τ the process is stopped and all subsequent values are set to $\tilde{q}_i^k(t) = 0$. With this definition (4.81) is equivalent to

$$\lim_{t\to\infty} \tilde{q}_i^k(t) = 0, \quad \text{a.s.,} \qquad \text{for all } k, i \neq o_k \tag{4.83}$$

I.e., the definition of the stopped process allows replacement of the limit infimum in (4.80) by a regular limit in (4.83).Without loss of generality we consider $\tau = 0$ in (4.82) to simplify notation.

Consider the queue update defined in equation (4.1) rewritten in terms of the routing function in equation (4.8),

$$q_i^k(t+1) = q_i^k(t) - \sum_{j\in n_i} \big(R_{ij}^k(\lambda_t) - R_{ji}^k(\lambda_t) - a_i^k(t)\big). \tag{4.84}$$

Define a σ-algebra encoding the histories for the queues and dual variables up to time t,

$$\sigma_t = \{\lambda_\tau, q_\tau \,|\, \forall \tau \leq t\}. \tag{4.85}$$

Taking expectation with respect to the arrival process and conditioned on the past history as encoded by σ_t,

$$\mathbb{E}\left[q_i^k(t+1) \,\middle|\, \sigma_t\right] = q_i^k(t) - \sum_{j\in n_i} \left(R_{ij}^k(\lambda_t) - R_{ji}^k(\lambda_t) - a_i^k\right) \tag{4.86}$$

where we recall $\mathbb{E}\left[a_i^k(t)\right] = a_i^k$. Observe from definition (4.9)

$$\bar{g}_i^k(\lambda_t) + \xi = \sum_{j\in n_i} \big(R_{ij}^k(\lambda_t) - R_{ji}^k(\lambda_t) - a_i^k\big). \tag{4.87}$$

Substituting into (4.86) we have

$$\mathbb{E}\left[q_i^k(t+1) \,\middle|\, \sigma_t\right] \leq q_i^k(t) - \bar{g}_i^k(\lambda_t) - \xi. \tag{4.88}$$

From proposition 7 we know that $\|\bar{g}(\lambda_t)\|$ converges to zero almost surely, therefore there exists a T such that $\|\bar{g}(\lambda_t)\| \leq \xi - \epsilon$ for all $t \geq T$ for any $\epsilon \in (0, \xi)$. For any particular (i, k), we have

$|\bar{g}_i^k(\lambda_t)| \leq \|\bar{g}(\lambda_t)\|$, allowing us to conclude that

$$\mathbb{E}\left[q_i^k(t+1) \,\middle|\, \sigma_t\right] \leq q_i^k(t) - \epsilon, \qquad \forall t \geq T \tag{4.89}$$

for all $q_i^k(t) \geq \epsilon$ because $q_i^k(t) \geq 0$ for all t.

Recalling the definition of the stopped process in (4.82) it follows from (4.89) that there exists a time T such that

$$\mathbb{E}\left[\tilde{q}_i^k(t+1) \,\middle|\, \sigma_t\right] < \tilde{q}_i^k(t) \tag{4.90}$$

for all $\tilde{q}_i^k(t) > 0$, therefore $\tilde{q}_i^k(t)$ converges to zero almost surely. While $q_i^k(t)$ may move away from zero after $\tilde{q}_i^k(t)$ converges, we reinitialize the stopped martingale process to show that $q_i^k(t)$ must eventually return to zero. ■

Corollary 1. Consider the Accelerated Backpressure algorithm, defined by (4.52) with $d_t^{(N)}$ as defined in (4.46) and primal Lagrangian maximizers defined in (4.10). If the step size sequence is chosen to satisfy $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$, all queues become empty infinitely often with probability one,

$$\liminf_{t\to\infty} q_i^k(t) = 0, \quad \text{a.s.,} \qquad \text{for all } k, i \neq o_k. \tag{4.91}$$

Proof : From Proposition 7, we know that $||\bar{g}(\lambda_t)||$ converges to zero which allows application of Proposition 8. From Proposition 8 it is guaranteed that each queue has a limit infimum equal to zero, which is equivalent to having each queue become empty infinitely often. ■

Corollary 1 is a sufficient condition for queue stability. In fact it is a much stronger result because the queues are always eventually cleared. This occurs because our dual variables converge to the optimal priorities which yield a routing $R_{ij}^k(\lambda_k)$ which forces the queue lengths to decrease in Expectation whenever they are non-zero.

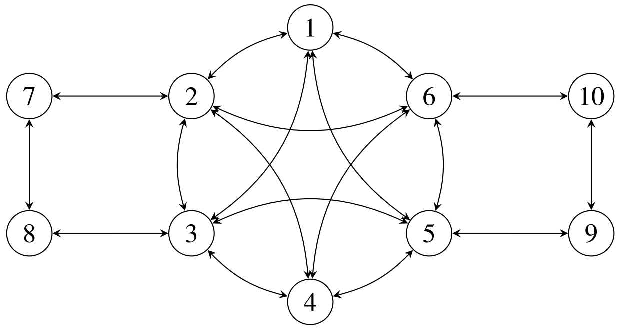


Figure 4.3: Several numerical experiments for the ABP algorithm presented in this section are performed on this 10 node network with 5 data types. The destinations are unique for each data type and are chosen randomly.

4.4 Numerical Experiments

Numerical experiments are performed to compare the ABP algorithm summarized in algorithms 4 and 5 with the Backpressure and Soft Backpressure algorithms summarized in algorithms 2 and 3. In all algorithms we use a fixed step size $\alpha_t = 1$. Observe that constant stepsizes are not covered by the guarantees of Section 4.3. We use constant stepsizes here to understand this different scenario. The objective function used for Soft Backpressure and ABP is of the form

$$f_{ij}^k(r_{ij}^k) = -\frac{1}{2}\left(r_{ij}^k\right)^2 + \beta_{ij}^k r_{ij}^k. \tag{4.92}$$

The quadratic terms increase the cost of routing a large number of packets across a single link and help to eliminate myopic routing choices that lead to sending packets in cycles. The linear terma β_{ij}^k are introduced to reward the transmission of packets to their final destinations. In our experiments we set $\beta_{ij}^k = 10$ for all edges routing to their respective data type destinations, i.e., when $j = o_k$, and set $\beta_{ij}^k = 0$ for all other i, j, k. With this choice for f_{ij}^k the derivative inverse function in (4.15) is simply $F_{ij}^k(y) = \beta_{ij}^k - y$. This simple derivative inverse function simplifies implementation of steps 5 and 6 of algorithms 5 and 3. In all algorithms we make $\xi = 0$ and for ABP we further set the approximation parameter to $N = 1$ so that ABP-1 is implemented. Of all the ABP algorithms, ABP-1 is the one with smallest communication overhead. Higher order members of the ABP family perform better than ABP-1.

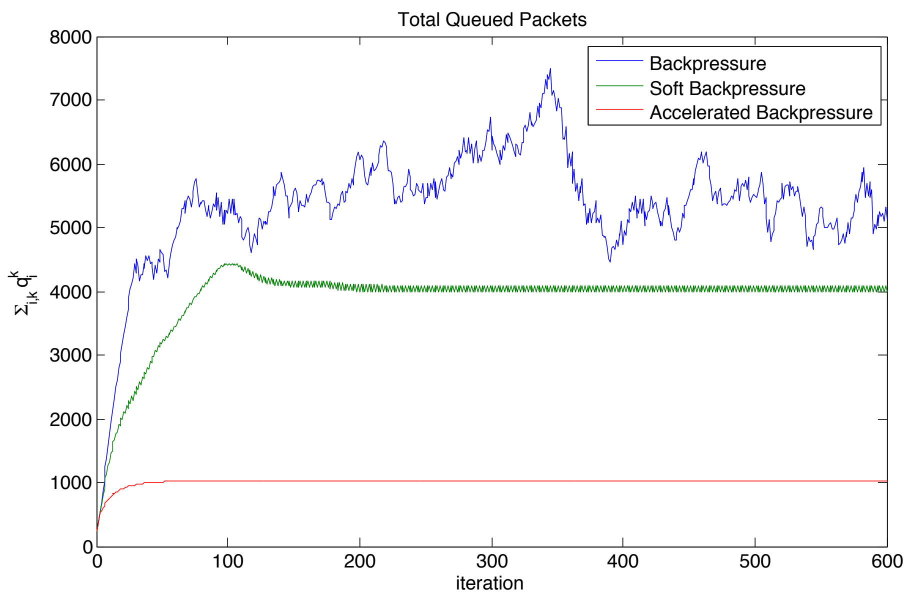


Figure 4.4: Without access to arrival statistic, the ABP algorithm learns a routing strategy that stabilizes the queues faster than the Soft Backpressure or Backpressure algorithms, leading to fewer queued packets at steady state.

We begin by considering the 10 node network shown in Figure 4.3 with individual link capacities C_{ij} chosen uniformly at random from the interval $[10, 100]$. We consider the problem of routing 5 data types having destinations that are chosen at random and average arrival rates of 5 packets per unit of time for each data type and node. This results in an average total load of 45 packets per flow and 225 packets per unit of time for the network as a whole. Recall that these arrival rates are unknown to nodes which respond to the number of packets that are received in each time slot. The number of packets that arrive in each time slot is selected from the discrete uniform distribution over the set $\{0, 1, \ldots, 10\}$.

Figure 4.4 shows the total number of packets queued in the system as a function of elapsed time for a sample run. ABP-1 stabilizes the queues after about 50 iterations while Soft Backpressure requires in excess of 100 iterations. The number of iterations that Backpressure needs to stabilize queues is difficult to judge due to its volatility but it seems that it takes in the order of 300 to 400 iterations for the queues to stop growing. Easier to judge is the fact that ABP-1 stabilizes queues at a much shorter length. The total number of packets queued never exceeds 1,110. Soft Backpressure

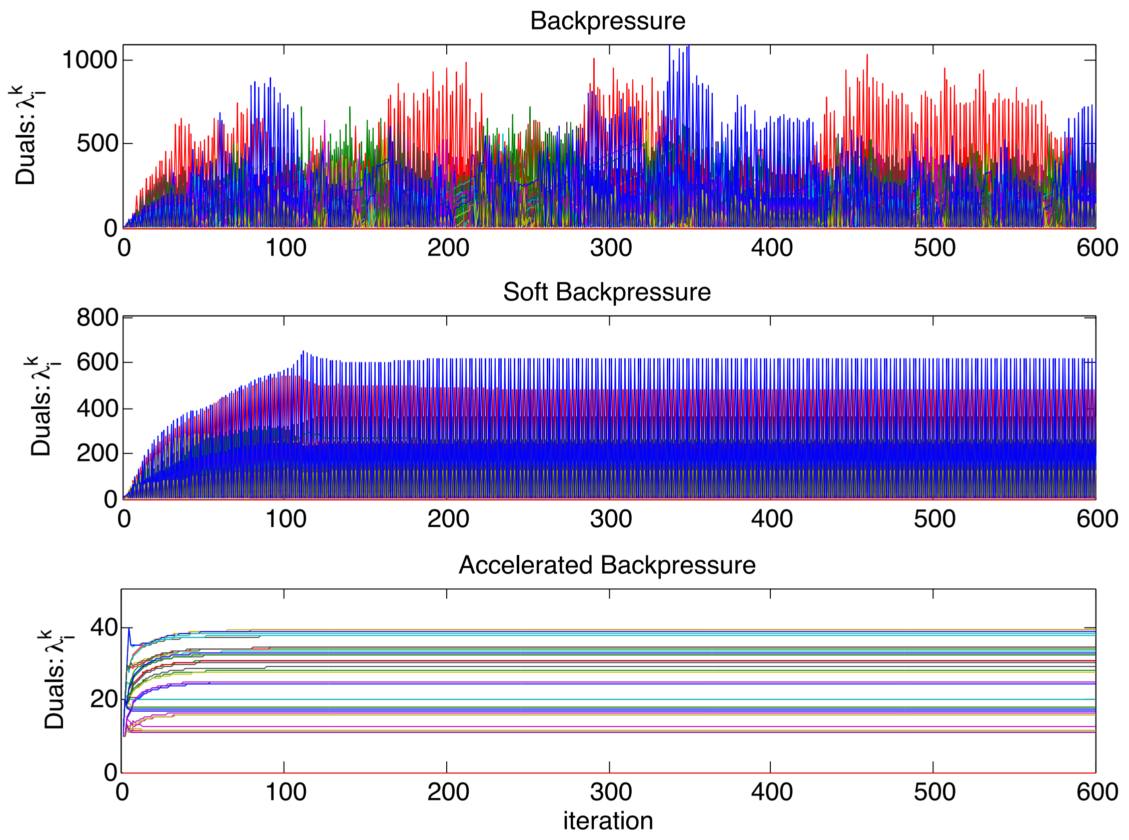


Figure 4.5: The Accelerated Backpressure algorithm converges to the optimal dual variables yielding a consistent routing strategy rather than the oscillatory solutions generated by Backpressure and Soft Backpressure.

stabilizes with around 4,000 packets in the system, while Backpressure grows to more than 7,000 queued packets before reducing the number of packets to about 5,000. Further observe that in addition to having shorter queues, ABP-1 has smaller variations in queue lengths.

The reason why ABP-1 is able to achieve less volatility in queue lengths is because the dual variables approach their optimal values, which in turn leads to less variation in the routing variables computed from them. This is not true of Backpressure and Soft Backpressure whose dual variables have large oscillations around their optimal values. This is illustrated in Figure 4.5. The dominant feature of dual variables for Backpressure and Soft Backpressure is their oscillatory behavior. For ABP-1 the dual variables converge to their optimal values in about 50 iterations, which explains why queues stabilize after this much time elapses.

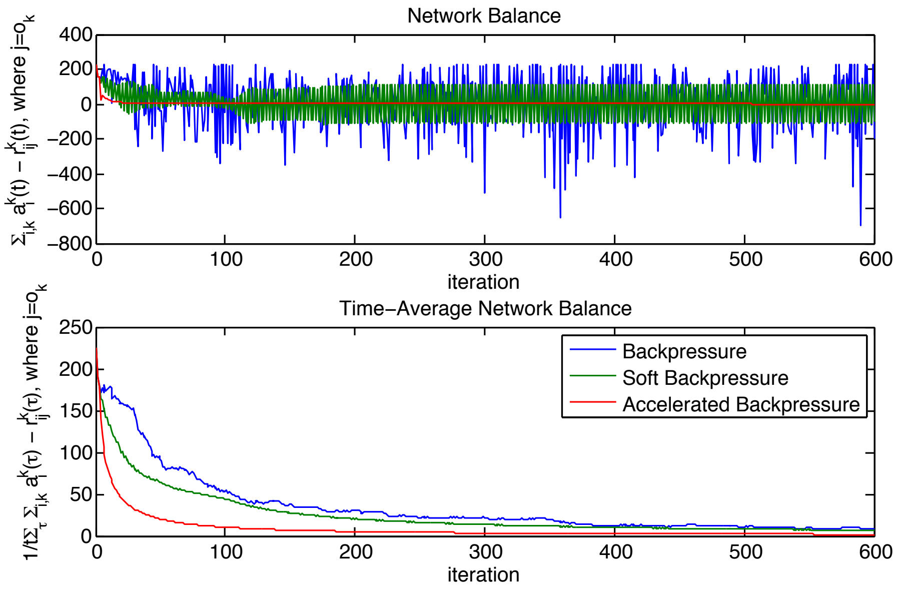


Figure 4.6: A successful routing strategy is characterized by routing as many packets out of the network as are arriving exogenously. While Soft Backpressure and soft Backpressure can achieve this state on average, Accelerated Backpressure achieves this balance on a per iteration basis.

The resulting stability in routing variables is illustrated in Figure 4.6-top. Shown in this figure is the instantaneous value of the queue imbalances $\sum_{j\in n_i} r_{ij}^k(t) - r_{ji}^k(t) - a_i^k(t)$ summed over the whole network. Observe how ABP-1 succeeds in satisfying these constraints for all times – this is not strictly true, there are still random variations around $\sum_{j\in n_i} r_{ij}^k(t) - r_{ji}^k(t) - a_i^k(t) = 0$ but they are small. For Backpressure and Soft Backpressure this is not true instantaneously. It is true on average, as we show in Figure 4.6-bottom, and this is sufficient to stabilize queues. However, the instantaneous variations in this constraints make the behavior of Backpressure and Soft Backpressure more erratic than the behavior of ABP-1. The cost of achieving this more stable behavior is the added communication overhead as summarized in Table 4.1.

The results in figures 4.4-4.6 are illustrative of a sample run. A more comprehensive analysis involves studying statistics across a number of realizations. We do so for the average queue balance constraints in Figure 4.7, which we can interpret as a statistical version of Figure 4.6-bottom. The averages in Figure 4.7 are across 1,000 trials. ABP-1 achieves queue balance after fewer than 50 iterations, while Soft Backpressure requires over 250 and Backpressure fails to achieve balance

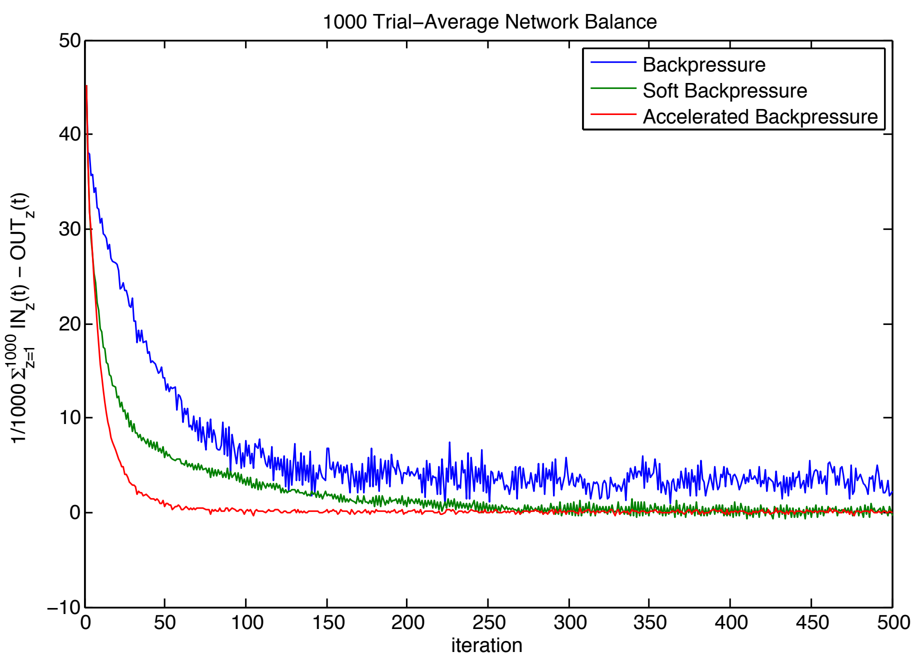


Figure 4.7: The rate of convergence for each algorithm is effectively how quickly it can achieve a network balance. Performance can be difficult to judge in a single realization so we compare the average across 1000 trials on barely feasible networks.

even after 500 iterations have passed. While there are small quantitative differences between figures 4.6-bottom and 4.7, their qualitative properties are the same.

Since network balance is equivalent to achieving a feasible solution to the primal optimization problem, it is interesting to consider what happens when the edge capacities are reduced, shrinking the set of feasible routing strategies. To evaluate the effect of edge capacities, we consider the queue stabilization problem with capacities $C = \rho C_0$ where $[C_0]_{ij}$ are the capacities for edge (i, j) and the capacity fraction ρ is halved until the problem becomes infeasible, i.e., $\rho = 1, 1/2, 1/4$, etc. Since the arrival rates are stochastic, the experiment is repeated 100 times for each ρ and the mean and standard deviation are shown in Figure 4.8. The results show that ABP has robust performance as the boundary of feasibility is approached. Error bars which show ± 1 standard deviation, indicate that the ABP and Soft Backpressure solutions are not significantly affected by the realization of packet arrivals until the problem is nearly infeasible. Backpressure solutions had standard deviation in excess of 50% of the mean, implying a significant dependence on the packet arrival realization.

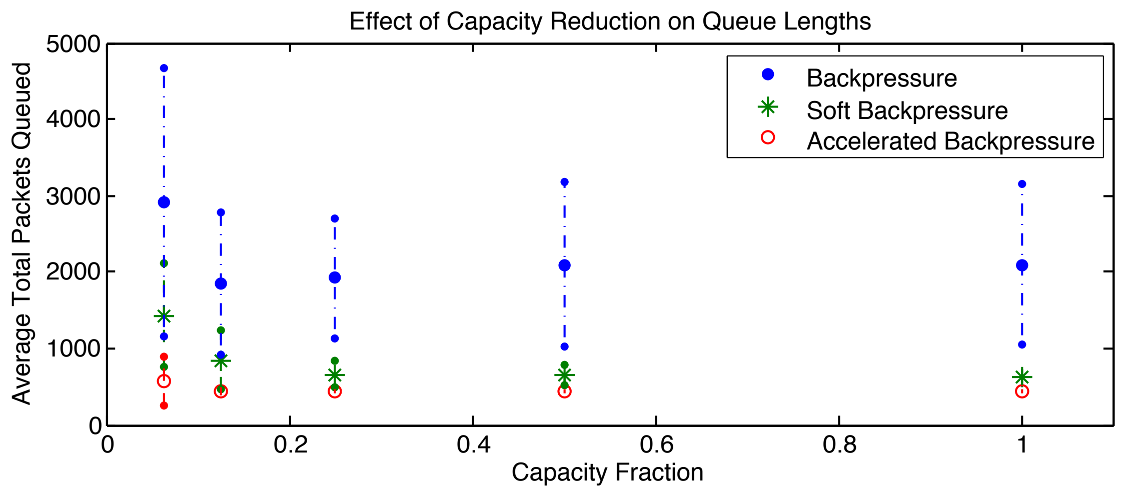


Figure 4.8: The total packets queued are averaged over 100 realizations of packet arrive sequences for each data point. The capacities of the edges are reduced by 50% and repeat the experiment until the problem becomes infeasible. Error bars show ± 1 standard deviation. Reduction in network capacity doesn't significantly effect performance until the problem is nearly infeasible.

The analysis that we made for the specific network topology in Figure 4.3 still hold for more general networks. To demonstrate this statement we consider randomly generated networks. The nodes are placed uniformly at random on $[0, 1] \times [0, 1]$ and are connected if they fall within a connectivity radius of 0.4 of each other. The arrival statistics, edge capacities and objective function remain as before. From Figure 4.9, it is observed that the ABP algorithm outperforms Soft Backpressure and Backpressure in trials on 100 randomly generated proximity networks with 20 nodes. ABP has fewer steady state queued packets than Soft Backpressure and Backpressure in every trial. The total number of packets queued at steady state using ABP is on average about 40% less than the number of packets queued by Backpressure. The Soft Backpressure algorithm occasionally performs as well as ABP, but on average ABP reduces the total queued packets by 25%.

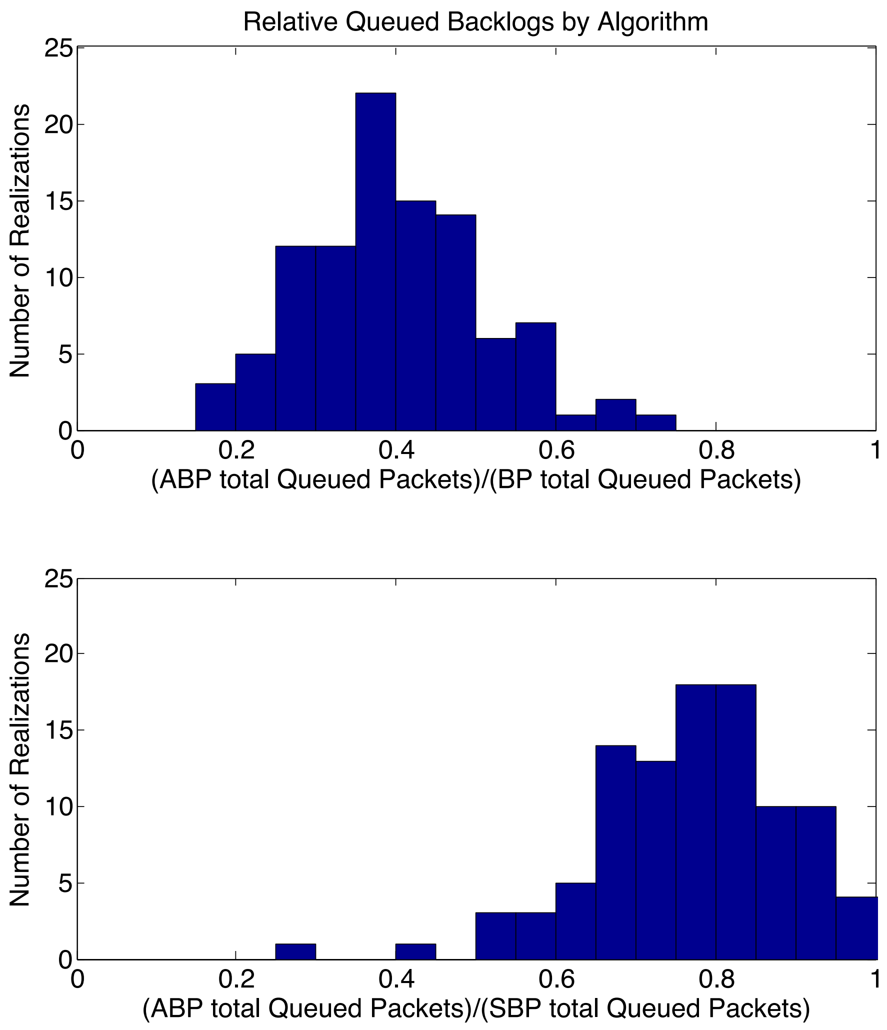


Figure 4.9: (top) Over 100 trials on random proximity networks, the total number of packets queued at steady state under the ABP algorithm is on average about 40% the number left in queue by the Backpressure algorithm. (bottom) The Soft Backpressure algorithm occasionally performs as well as Accelerated Backpressure but on average Accelerated Backpressure reduces the steady state queued packets by about 25%.

Chapter 5

Discounted Integral Priority Routing

Chapter 5 also considers the problem of joint routing and scheduling in packet networks. As in Chapter 4, packets are accepted from upper layers as they are generated and marked for delivery to intended destinations. To accomplish delivery of information nodes need to determine routes and schedules capable of accommodating the generated traffic. From a node-centric point of view, individual nodes handle packets that are generated locally as well as packets received from neighboring nodes. The goal of each node is to determine suitable next hops for each flow conducive to successful packet delivery.

The study of the joint scheduling and routing problem, has been built up from the Backpressure (BP) algorithm, [Tassiulas and Ephremides, 1992] and the soft backpressure algorithm (SBP), [Ribeiro, 2009b]. In [Ribeiro, 2009b], Soft Backpressure (SBP) includes an objective function in the feasibility problem. For appropriate choice of objective the subgradient becomes a unique gradient, improving convergence. In, [Zargham et al., 2013a]. Accelerated Backpressure (ABP) extends this idea of SBP further by choosing a strongly convex objective and implementing Accelerated Dual Descent (ADD), a distributed approximation to Newton's Method in place of gradient descent, [Zargham et al., 2014c]. SBP and ABP solve these stabilization problems much faster than BP but these algorithms but are unable to effectively clear backlogged queues even if the capacity to do so is available in the network.

In this chapter, we shift into more control oriented thinking and observe that in order to eliminate large queues in steady state an integral control term is necessary. In developing Discounted Integral Priority (DIP) routing, we work within the Soft Backpressure framework but rather than using the queues themselves as routing priorities, we use a time discounted sum of the queue history. These priorities can be computed locally using a simple linear update combining the existing priorities and the newly observed queue lengths. Our method can be connected to a class of heavy ball methods like those introduced in [Nesterov, 1983]. By recasting the Discounted Integral Priority routing as a stochastic heavy ball update for the dual variables, we are able to implement a decaying step size which is a standard requirement for convergence in stochastic optimization. Our stability proofs leverage the decaying step size to prove queue stability under the stochastic heavy ball variant of the DIP algorithm. One of the main challenges in reducing the queues rather than just stabilizing them is the stochastic nature of the packet arrivals. Some related work includes noise cancellation in [Beck and Teboulle, 2009b] and adaptive gradient methods in [Duchi et al., 2011].

5.1 Packet Routing Problem Revisited

5.1.1 Problem Formulation

Consider a given network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of links between nodes. Denote as C_{ij} the capacity of link $(i, j) \in \mathcal{E}$ and define the neighborhood of i as the set $n_i = \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$ of nodes j that can communicate directly with i. There is also a set of information flows $\mathcal{K}$ with the destination of flow $k \in \mathcal{K}$ being the node $o_k \in \mathcal{V}$. Let $n = |\mathcal{V}|$ be the number of nodes and $K = |\mathcal{K}|$ be the number of information flows in the network and $E = |\mathcal{E}|$ be the number of edges in the network. Define the $n \times E$ matrix A to be the incidence matrix of the graph $\mathcal{G}$ and the reduced incidence matrix A_k as the $(n-1) \times E$ matrix with the row associated with destination node o_k removed. The block diagonal matrix $\bar{A}$ =diag$[A_k]$ is an $(n-1)K \times E \cdot K$ incidence matrix encoding the interrelation of all information flows and nodes in the network.

At time index t terminal $i \neq o_k$ generates a random number

$$a_i^k(t) = a_i^k + \nu_i^k(t) \tag{5.1}$$

of units of information to be delivered to o_k. The random variables $a_i^k(t) \geq 0$ are generated by $\nu_i^k(t)$ which are independent and identically distributed across time with $\mathbb{E}\left[\nu_i^k(t)\right] = 0$ and finite support. Thus the expected value $\mathbb{E}\left[a_i^k(t)\right] = a_i^k$ and $a_i^k(t)$ has finite support. In time slot $[t, t+1)$, node i routes $r_{ij}^k(t) \geq 0$ units of information through neighboring node $j \in n_i$ and receives $r_{ji}^k(t) \geq 0$ packets from neighbor j. The difference between the total number of received packets $a_i^k(t) + \sum_{j \in n_i} r_{ji}^k(t)$ and the sum of transmitted packets $\sum_{j \in n_i} r_{ij}^k(t)$ is added to the local queue – or subtracted if this quantity is negative. Therefore, the number $q_i^k(t)$ of k-flow packets queued at node i evolves according to

$$q_i^k(t+1) = \left[q_i^k(t) + a_i^k(t) + \sum_{j \in n_i} r_{ji}^k(t) - r_{ij}^k(t)\right]^+, \tag{5.2}$$

where the projection $[\cdot]^+$ into the nonnegative reals is necessary because the number of packets in queue cannot become negative. We remark that (5.2) is stated for all nodes $i \neq o_k$ because packets routed to their destinations are removed from the system.

To ensure packet delivery it is sufficient to guarantee that all queues $q_i^k(t+1)$ remain stable. In turn, this can be guaranteed if the average rate at which packets exit queues does not exceed the rate at which packets are loaded into them. To state this formally observe that the time average limit of arrivals satisfies $\lim_{t \to \infty} a_i^k(t) = \mathbb{E}\left[a_i^k(t)\right] := a_i^k$ and define the ergodic limit $r_{ij}^k := \lim_{t \to \infty} r_{ij}^k(t)$. If the processes $r_{ij}^k(t)$ controlling the movement of information through the network are asymptotically stationary, queue stability follows if

$$\sum_{j \in n_i} r_{ij}^k - r_{ji}^k \geq a_i^k + \xi \quad \forall\, k, i \neq o_k \tag{5.3}$$

for some constant $\xi > 0$ which is introduced because stability is guaranteed if the inequalities

$\sum_{j\in n_i} r_{ij}^k - r_{ji}^k > a_i^k$ hold in a strict sense. For future reference define the vector $r := \{r_{ij}^k\}_{k,(i,j)}$ grouping variables r_{ij}^k for all information flows and links. Since at most C_{ij} packets can be transmitted in link (i,j) the routing variables $r_{ij}^k(t)$ always satisfy the capacity constraints on the network,

$$\sum_k r_{ij}^k(t) \leq C_{ij} \tag{5.4}$$

which defines the the set of possible routings

$$\mathcal{C} = \{r \in \mathbb{R}_+^{K\cdot E} : r_{ij}^k(t) \leq C_{ij} \forall (i,j) \in \mathcal{E}\}. \tag{5.5}$$

The joint routing and scheduling problem can be now formally stated as the determination of non-negative variables $r(t) \in \mathcal{C}$ that satisfy (5.4) for all times t and whose time average limits r_{ij}^k satisfy (5.3). The BP algorithm solves this problem by assigning all the capacity of the link (i,j) to the flow with the largest queue differential $q_i^k(t) - q_j^k(t)$. Specifically, for each link we determine the flow pressure

$$k_{ij}^* = \underset{k}{\operatorname{argmax}} \left[q_i^k(t) - q_j^k(t)\right]^+ . \tag{5.6}$$

If the maximum pressure $\max_k \left[q_i^k(t) - q_j^k(t)\right]^+ > 0$ is strictly positive we set $r_{ij}^k(t) = C_{ij}$ for $k = k_{ij}^*$. Otherwise the link remains idle during the time frame. The backpressure algorithm works by observing the queue differentials on each link and then assigning the capacity for each link to the data type with the largest positive queue differential, thus driving the time average of the queue differentials to zero– stabilizing the queues. To generalize, we reinterpret BP as a dual stochastic subgradient descent.

5.1.2 Dual Stochastic Subgradient Descent

Since the parameters that are important for queue stability are the time averages of the routing variables $r_{ij}^k(t)$ an alternative view of the joint routing and scheduling problem is the determination of variables r_{ij}^k satisfying (5.3) and $\sum_k r_{ij}^k \leq C_{ij}$. This can be formulated as the solution of an optimization problem. Let $f_{ij}^k(r_{ij}^k)$ be any concave function on $\mathbb{R}_+$ and consider the optimization problem

$$\begin{aligned} r^* := \operatorname{argmax} & \sum_{k,i\neq o_i,j} f_{ij}^k(r_{ij}^k) \\ \text{s.t.} \quad & \sum_{j\in n_i} r_{ij}^k - r_{ji}^k \geq a_i^k + \xi, \quad \forall\, k, i \neq o_k, \\ & \sum_{k\in\mathcal{K}} r_{ij}^k \leq C_{ij}, \qquad \forall\, (i,j) \in \mathcal{E}. \end{aligned} \tag{5.7}$$

where the domain is a the convex polyhedron defined in (5.5). Since only feasibility is important for queue stability, solutions to (5.7) ensure stable queues irrespectively of the objective functions $f_{ij}^k(r_{ij}^k)$. For notational compactness, define $f(r) = \sum_{k,(i,j)} f_{ij}^k(r_{ij}^k)$.

Since the problem in (5.7) is concave it can be solved by descending on the dual domain. Start by associating multipliers λ_i^k with the constraint $\sum_{j\in n_i} r_{ij}^k - r_{ji}^k \geq a_i^k$ and keep the constraint $r \in \mathcal{C}$ implicit. The corresponding Lagrangian associated with the optimization problem in (5.7) is

$$\mathcal{L}(r,\lambda) = \sum_{k,i\neq o_k,j} f_{ij}^k(r_{ij}^k) + \sum_{k,i\neq o_k} \lambda_i^k \left(\sum_{j\in n_i} r_{ij}^k - r_{ji}^k - a_i^k + \xi \right) \tag{5.8}$$

where we introduced the vector $\lambda := \{\lambda_i^k\}_{k,i\neq o_k}$ grouping variables λ_i^k for all flows and nodes. The corresponding dual function is defined as

$$h(\lambda) := \max_{r\in\mathcal{C}} \mathcal{L}(r,\lambda). \tag{5.9}$$

To compute a descent direction for $h(\lambda)$, compute the primal Lagrangian maximizers for given λ according to the vector function $R : \mathbb{R}_+^{(n-1)K} \to \mathcal{C}$ defined

$$R(\lambda) := \operatorname*{argmax}_{r\in\mathcal{C}} \mathcal{L}(r,\lambda). \tag{5.10}$$

The individual elements $R_{ij}^k(\lambda)$ are ordered by stacking the E dimensional subvectors for each information flow k. A descent direction for the dual function is available in the form of the dual subgradient[1] are obtained by evaluating the constraint slack associated with the Lagrangian maximizers

$$[\nabla h(\lambda)]_i^k := \sum_{j \in n_i} R_{ij}^k(\lambda) - R_{ji}^k(\lambda) - a_i^k - \xi. \tag{5.11}$$

Since the Lagrangian $\mathcal{L}(r, \lambda)$ in (5.8) is linear in the dual variables λ_i^k the determination of the maximizers $R_{ij}^k(\lambda) := \operatorname{argmax}_{r \in \mathcal{C}} \mathcal{L}(r, \lambda)$ can be decomposed into the maximization of separate summands. Considering the coupling constraints $\sum_k r_{ij}^k \leq C_{ij}$ imposed by the domain $\mathcal{C}$ it suffices to consider variables $\{r_{ij}^k\}_k$ for all flows across a given link. After reordering terms it follows that we can compute routes link wise

$$\begin{aligned} R_{ij}^k(\lambda) = &\operatorname*{argmax}_{\{r_{ij}^k \geq 0\}_k} \sum_k f_{ij}^k(r_{ij}^k) + r_{ij}^k\left(\lambda_i^k - \lambda_j^k\right) \\ &\text{s.t.} \qquad \sum_{k \in \mathcal{K}} r_{ij}^k \leq C_{ij} \end{aligned} \tag{5.12}$$

for each link $(i, j) \in \mathcal{E}$. Introducing a time index t, subgradients $\nabla h_i^k(\lambda_t)$ could be computed using (5.11) with Lagrangian maximizers $R_{ij}^k(\lambda_t)$ given by (5.12). For notational convenience, the dual subgradient may also be specified in vector form

$$\nabla h(\lambda) = \bar{A} R(\lambda) - a - \xi \mathbf{1} \tag{5.13}$$

where $\mathbf{1}$ is the vector of all ones. A subgradient descent iteration could then be defined to find the variables r^* that solve (5.7) via a dual method which generates a sequence λ_t; see e.g., [Ribeiro and Giannakis, 2007].

The problem in computing $\nabla h_i^k(\lambda)$ is that we don't know the average arrival rates a_i^k. We do

[1]For an appropriately chosen cost function $f(r)$, the subgradient is unique which motives the use of gradient notation $\nabla h(\lambda)$, (see Proposition 9).

observe, however, the instantaneous rates $a_i^k(t)$ that are known to satisfy $\mathbb{E}\left[a_i^k(t)\right] = a_i^k$. Therefore,

$$[g_t(\lambda)]_i^k := \sum_{j \in n_i} R_{ij}^k(\lambda) - R_{ji}^k(\lambda) - a_i^k(t) - \xi, \tag{5.14}$$

is a stochastic subgradient of the dual function in the sense that its expected value $\mathbb{E}\left[g_i^k(\lambda)\right] = \nabla h_i^k(\lambda)$ is the subgradient defined in (5.11). Stated in vector form

$$g_t(\lambda) = \bar{A}R(\lambda) - a - \nu_t - \xi\mathbf{1}, \tag{5.15}$$

where $a_t = a + \nu_t$ and $\mathbb{E}[\nu_t] = 0$ for all t. We can then minimize the dual function using a stochastic subgradient descent algorithm. At time t we have multipliers λ_t and determine Lagrangian maximizers $r_{ij}^k(t) := R_{ij}^k(\lambda_t)$ as per (5.12). We then proceed to update multipliers along the stochastic subgradient direction according to

$$\lambda_i^k(t+1) = \left[\lambda_i^k(t) - \epsilon\Bigg(\sum_{j \in n_i} r_{ij}^k(t) - r_{ji}^k(t) - a_i^k(t) - \xi\Bigg)\right]^+, \tag{5.16}$$

where ϵ is a constant stepsize chosen small enough so as to ensure convergence; see e.g., [Ribeiro, 2009b].

Properties of the descent algorithm in (5.16) vary with the selection of the functions $f_{ij}^k(r_{ij}^k)$. Two cases of interest are when $f_{ij}^k(r_{ij}^k) = 0$ and when $f_{ij}^k(r_{ij}^k)$ are continuously differentiable, strongly convex, and monotone decreasing on $\mathbb{R}_+$ but otherwise arbitrary. The former allows recovers the Backpressure Algorithm while the latter leads to the Soft Backpressure algorithm.

5.1.3 Soft Backpressure

Assume now that the functions $f_{ij}^k(r_{ij}^k)$ are continuously differentiable, strongly convex, and monotone decreasing on $\mathbb{R}_+$ but otherwise arbitrary. In this case the derivatives $\partial f_{ij}^k(x)/\partial x$ of the functions $f_{ij}^k(x)$ are monotonically increasing and thus have inverse functions that we denote as

$$F_{ij}^k(x) := \left[\partial f_{ij}^k(x)/\partial x\right]^{-1}(x). \tag{5.17}$$

The Lagrangian maximizers in (5.12) can be explicitly written in terms of the derivative inverses $F_{ij}^k(x)$. Furthermore, the maximizers are unique for all λ implying that the dual function is differentiable. The details are outlined in Proposition 9 originally published in [Zargham et al., 2013a].

Proposition 9. If the functions $f_{ij}^k(r_{ij}^k)$ in (5.7) are continuously differentiable, strongly concave, and monotone decreasing on $\mathbb{R}_+$, the dual function $h(\lambda) := \max_{r\in\mathcal{C}} \mathcal{L}(r,\lambda)$ is differentiable for all λ. Furthermore, the gradient component along the λ_i^k direction is $[\nabla h(\lambda)]_i^k$ as defined in (5.11) with

$$R_{ij}^k(\lambda) = F_{ij}^k\left(-\left[\lambda_i^k - \lambda_j^k - \mu_{ij}(\lambda)\right]^+\right), \tag{5.18}$$

where $\mu_{ij}(\lambda)$ is either 0 if $\sum_k F_{ij}^k\big(-\left[\lambda_i^k - \lambda_j^k\right]^+\big) \leq C_{ij}$ or chosen as the solution to the equation

$$\sum_k F_{ij}^k\left(-\left[\lambda_i^k - \lambda_j^k - \mu_{ij}(\lambda)\right]^+\right) = C_{ij}. \tag{5.19}$$

Proof : This proposition is equivalent to Proposition 4. See the Proof in Chapter 4. ■

While (5.19) does not have a closed for solution it can be computed quickly numerically using a binary search because it is a simple single variable root finding problem. Computation time cost remains small compared to communication time cost.

The Soft backpressure can be implemented using node level protocols. At each time instance nodes send their multipliers $\lambda_i^k(t)$ to their neighbors. After receiving multiplier information from its neighbors, each node can compute the multiplier differentials $\lambda_i^k(t)-\lambda_j^k(t)$ for each edge. The nodes then solve for μ_{ij} on each of its outgoing edges by using a rootfinder to solve the local constraint in (5.19), The capacity of each edge is then allocated to the unique information flows via reverse waterfilling as defined in (5.18). Once the transmission rates are set each node can observe its net packet gain which is equivalent to the stochastic gradient as defined in (5.14). Finally, each node updates its multipliers by subtracting ϵ times the stochastic subgradient from its current multipliers. As with BP, choosing the stepsize $\epsilon = 1$ causes the multipliers to coincide with the queue lengths for all time.

5.1.4 A General Priority-Based Routing Strategy

The Backpressure and and Soft Backpressure algorithms, as detailed in the preceding sections, are fundamentally priority based routing strategies. When viewed in the optimization framework these priorities are dual variables but the priority-based routing strategy $R(\lambda)$ defined in (5.10) can be implemented for any priority vector λ. We define a general priority-based routing strategy

$$r_t = R(\lambda_t) \text{ for any sequence } \lambda_t,\ \forall t. \tag{5.20}$$

A priority based routing strategy based on (5.20) generates a sequence of queue lengths

$$q_{t+1} = q_t - g_t(\lambda_t) - \xi \mathbf{1} \tag{5.21}$$

where the stochastic gradient function, $g_t(\lambda)$ is defined in (5.14). The $\xi\mathbf{1}$ offset guarantees that priorities λ_t satisfying $g_t(\lambda_t) \geq 0$ result in a strict reduction. Substituting the definition of the stochastic gradient

$$q_{t+1} = q_t - \bar{A}R(\lambda_t) + a + \nu_t. \tag{5.22}$$

We define the queue update

$$\Delta q_t(\lambda_t) = \bar{A}R(\lambda_t) - a - \nu_t. \tag{5.23}$$

The routing strategy $R(\lambda_t)$ for any sequence $\{\lambda_t\}_{t=0}^{\infty}$ generalizes the notion of Soft Backpressure, [Ribeiro, 2009b] to a case where the priorities can be generated by any desirable scheme. Another example of a priority based routing algorithm is Accelerated Backpressure, [Zargham et al., 2013a]. Priority sequences $\{\lambda_t\}_{t=0}^{\infty}$ are not all equally effective. Determination of sufficient condi-

Algorithm 6: Priority-Based Routing at node i

```
1 for $t = 0, 1, 2, \cdots$ do
2     Observe or Compute Priorities $\{\lambda_i^k(t)\}_k$
3     for all neighbors $j \in n_i$ do
4         Send priorities $\{\lambda_i^k(t)\}_k$ – Receive priorities $\{\lambda_j^k(t)\}_k$
5         Compute $\mu_{ij}$ such that
              $\sum_k [\lambda_i^k(t) - \lambda_j^k(t) - \mu_{ij}]^+$
          Transmit packets at rate
              $r_{ij}^k(t) = F(-[\lambda_i^k(t) - \lambda_j^k(t) - \mu_{ij}]^+)$
6     end
7 end
```

tions for a priority sequence $\{\lambda_t\}_{t=0}^{\infty}$ to yield stable queue lengths is an open problem.

5.2 Discounted Integral Priority Based Routing

Observing that existing protocols for packet forwarding which use optimization motivated priorities tend to stabilize but not reduce the queues, we propose a set of priorities motivated by integral control. In particular, since the queues are state variables which we would like to make small at steady state, we set the priorities to a discounted integral of the queues. Consider the set of routing priorities

$$\lambda_t = \sum_{\tau=0}^{t} \alpha^{t-\tau} q_t \tag{5.24}$$

where $\alpha \in [0, 1)$ is a discounting factor. Observe that for if $\alpha = 0$, the soft backpressure algorithm $\lambda_t = q_t$ is trivially recovered. Backpressure type algorithms are implemented by observing the stochastic gradients $g_t(\lambda_t)$ in order to update the routing variables. In our method, q_t is observed and the priorities are updated

$$\lambda_t = \alpha\lambda_{t-1} + q_t, \tag{5.25}$$

which can be computed without information from neighboring nodes: $\lambda_i^k(t) = \alpha\lambda_i^k(t-1) + q_i^k(t)$ for all nodes i, information flows k and times t. Under this scheme we do not force any information

Algorithm 7: Discounted Integral Priority Based Routing

```
1 for $t = 0, 1, 2, \cdots$ do
2     Observe $\{q_i^k(t)\}_k$,
3     Compute $\lambda_k^i(t) = \alpha\lambda_i^k(t-1) + q_i^k(t)$
4     for all neighbors $j \in n_i$ do
5         Send priorities $\{\lambda_i^k(t)\}_k$ – Receive priorities $\{\lambda_j^k(t)\}_k$
6         Compute $\mu_{ij}$ such that
              $$\sum_k F\left(-[\lambda_i^k(t) - \lambda_j^k(t) - \mu_{ij}]^+\right) = C_{ij}$$
          Transmit packets at rate
              $$r_{ij}^k(t) = F(-[\lambda_i^k(t) - \lambda_j^k(t) - \mu_{ij}]^+)$$
7     end
8 end
```

exchange, rather we allow information to spread through the effect of the changes in the realized routing. See Algorithm 7 for the distributed implementation of the priority based routing $r_t = R(\lambda_t)$ using the discounted integral priorities $\{\lambda_t\}_{t=0}^{\infty}$ chosen according to equation (5.24).

5.2.1 Connection to Heavy Ball Methods

The discounted integral priority update can be expressed as a variant on the heavy ball method. A variety of closely related momentum based algorithms are considered *heavy Ball Methods*, the primary examples are Nesterov's accelerated method, [Nesterov, 1983] and Polyak's Method, [Polyak, 1964].

Lemma 11. The discounted Integral Priority update in (5.25) can be rewritten as

$$\lambda_{t+1} = \lambda_t + \Delta q_t(\lambda_t) + \alpha(\lambda_t - \lambda_{t-1}) \tag{5.26}$$

where $\Delta q_t(\lambda_t)$ is the change in queues induced by routing according to the priorities λ_t and the discounting parameter α is weighting coefficient on the momentum term.

Proof : Stating equation (5.25) at time $t+1$ yields

$$\lambda_{t+1} = \alpha\lambda_t + q_{t+1}. \tag{5.27}$$

Subtracting (5.25) from (5.27) yields

$$\lambda_{t+1} = \lambda_t + (q_{t+1} - q_t) + \alpha(\lambda_t - \lambda_{t-1}). \tag{5.28}$$

From the definitions in (5.22) and (5.23), the change in the queues $\Delta q_t(\lambda) = q_{t+1} - q_t$ completing the proof. ■

From Lemma 11, we see that the discounted integral priority method is equivalent to applying the heavy ball method to a basic soft backpressure algorithm. Soft back pressure in our notation is $q_{t+1} = q_t - \bar{A}R(q_t) + a + \nu_t$, because the priorities and the queues are equivalent for all t, [Ribeiro, 2009b]. Unlike the basic soft backpressure algorithm the queues and the priorities do not remain equivalent. Heavy ball type methods are known to significantly improve convergence rates.

5.2.2 Stochastic Heavy Ball Routing

The heavy ball method is analyzed for stochastic optimization in [Flam, 2004]. In order to guarantee convergence in stochastic optimization, it is necessary to introduce a decaying step size ϵ_t satisfying $\sum_{t=1}^{\infty} \epsilon_t = \infty$ and $\sum_{t=1}^{\infty} \epsilon_t^2 < \infty$. Motivated by Lemma 11, we introduce Stochastic Heavy Ball routing

$$\lambda_{t+1} = \lambda_t - \epsilon_t g_t(\lambda_t) + \alpha(\lambda_t - \lambda_{t-1}) \tag{5.29}$$

which is a variant of the Discounted Integral Priority routing algorithm, which allows implementation of a decaying step size ϵ_t. This minor variation allows the direct application of Theorem 2 from [Flam, 2004] from which we conclude that λ_t converges with probability one, to a solution of the dual problem in (5.9). In the following section, it is shown that convergence of the priorities is sufficient to stabilize the queues.

5.3 Stability Analysis

The heavy ball variant of the discounted integral priority method with update as defined in (5.29) allows in the inclusion of a decaying step size which is standard in the analysis of stochastic optimization algorithms. Another key property is the boundedness of the stochastic gradient.

Lemma 12. The stochastic gradient is bounded

$$\|g_t(\lambda)\| \leq \gamma, \qquad \forall t,\, \forall \lambda. \tag{5.30}$$

Proof : Consider (5.14) and recall our assumption that the random portion of the arrival process, ν_t has finite support. Then observing from (5.10) that $R(\lambda)$ lies in a compact polyhedral set

$$R(\lambda) \in \mathcal{C} = \{r \geq 0 : \sum_k r_{ij}^k \leq C_{ij},\, \forall (i,j) \in \mathcal{E}\}, \tag{5.31}$$

implies that $g_t(\lambda)$ has a finite upper bound γ that holds for all λ and all t. ■

The analysis that follows is related to the stochastic convergence analysis for the Accelerated Backpressure (ABP) algorithm as detailed in [Zargham et al., 2014b]. As in ABP, the results are built on a specialized version of the supermartingale convergence theorem found in Theorem E.7.4 of [Solo and Kong, 1995]. The proofs are noticeably simpler than there ABP counterparts because there is no need to characterize the second derivatives of the dual objective.

Proposition 10. Consider the Stochastic Heavy Ball Routing implemented with the dual step (5.29), decaying step size ϵ_t satisfying $\sum_t \epsilon_t = \infty$, $\sum_t \epsilon_t^2 < \infty$ and the stochastic gradient $g_t(\lambda_t)$ as defined in (5.14), then

$$\lim_{t\to\infty} ||\nabla h(\lambda_t)\| = 0 \tag{5.32}$$

almost surely.

Proof : Define the unconstrained dual optimization problem

$$\min_{\lambda} h(\lambda) \tag{5.33}$$

where $h(\lambda)$ is the dual objective defined in (5.9). Theorem 2 of [Flam, 2004] defines the following convergence property for the stochastic heavy ball update defined in (5.29) with the decaying step size ϵ_t: λ_t converges almost surely to some $\hat{\lambda}$ satisfying

$$\hat{\lambda} = \mathcal{P}\left[\hat{\lambda} - \epsilon \nabla h(\hat{\lambda})\right] \qquad \forall \epsilon > 0 \tag{5.34}$$

where $\mathcal{P}[\cdot]$ is a projection onto the feasible set of (5.33). In this application, (5.33) is unconstrained, making $\mathcal{P}[\cdot]$ the identity map. Therefore, (5.34) simplifies to

$$\epsilon \nabla h(\hat{\lambda}) = 0 \qquad \forall \epsilon > 0. \tag{5.35}$$

guaranteeing that $\nabla h(\hat{\lambda}) = 0$. Using the convergence of λ_t to $\hat{\lambda}$, we have

$$\lim_{t \to \infty} ||\nabla h(\lambda_t)\| = ||\nabla h(\hat{\lambda})|| \tag{5.36}$$

completing the proof. ■

Proposition 10 is the first step in our theoretic convergence guarantee. By implementing a decaying step size when updating the dual variables, convergence to the optimal dual variables is achieved. From Theorem 2 of [Flam, 2004], priority vectors λ_t converge with probability one and our routing $R_{ij}^k(\lambda_t)$ becomes a feasible routing in the primal problem, (5.7). We proceed to leverage this fact to ensure the queues remain stable.

Proposition 11. Consider a dual variables process λ_t such that the dual gradient $\nabla h(\lambda_t)$ satisfies

$$\lim_{t \to \infty} ||\nabla h(\lambda_t)|| = 0, \quad \text{a.s.} \tag{5.37}$$

Then, all queues empty infinitely often with probability one, i.e.,

$$\liminf_{t\to\infty} q_i^k(t) = 0, \quad \text{a.s.}, \qquad \text{for all } k, i \neq o_k \tag{5.38}$$

Proof : The result in (5.38) is true if for arbitrary time $\tau \geq 0$ and arbitrary constant κ the queue length $q_i^k(t)$ is almost surely smaller than κ at some point the future. To write this statement formally define the stopped process

$$\tilde{q}_i^k(t) = \begin{cases} q_i^k(t) & \text{if } q_i^k(u) > 0 \text{ for all } \tau \leq u \leq t, \\ 0 & \text{else .} \end{cases} \tag{5.39}$$

The stopped process $\tilde{q}_i^k(t)$ follows the queue length process $q_i^k(t)$ until the queue length becomes null for the first time after time τ. If and when the queue empties after time τ the process is stopped and all subsequent values are set to $\tilde{q}_i^k(t) = 0$. With this definition (5.38) is equivalent to

$$\lim_{t\to\infty} \tilde{q}_i^k(t) = 0, \quad \text{a.s.}, \qquad \text{for all } k, i \neq o_k \tag{5.40}$$

I.e., the definition of the stopped process allows replacement of the limit infimum in (5.37) by a regular limit in (5.40).Without loss of generality we consider $\tau = 0$ in (5.39) to simplify notation.

Consider now the queue update defined in equation (5.2) rewritten in terms of the routing function in equation (5.10),

$$q_i^k(t+1) = q_i^k(t) - \sum_{j\in n_i} \big(R_{ij}^k(\lambda_t) - R_{ji}^k(\lambda_t) - a_i^k(t)\big). \tag{5.41}$$

Define a σ-algebra encoding the histories for the queues and dual variables up to time t,

$$\sigma_t = \{\lambda_\tau, q_\tau \,|\, \forall \tau \leq t\}. \tag{5.42}$$

Taking expectation with respect to the arrival process and conditioned on the past history as encoded by σ_t,

$$\mathbb{E}\left[q_i^k(t+1) \,\middle|\, \sigma_t\right] \leq q_i^k(t) - \sum_{j\in n_i} \left(R_{ij}^k(\lambda_t) - R_{ji}^k(\lambda_t) - a_i^k\right) \tag{5.43}$$

where we recall $\mathbb{E}\left[a_i^k(t)\right] = a_i^k$. Observe from definition (5.11)

$$\bar{g}_i^k(\lambda_t) + \xi = \sum_{j\in n_i} \left(R_{ij}^k(\lambda_t) - R_{ji}^k(\lambda_t) - a_i^k\right). \tag{5.44}$$

Substituting into (5.43) we have

$$\mathbb{E}\left[q_i^k(t+1) \,\middle|\, \sigma_t\right] \leq q_i^k(t) - \bar{g}_i^k(\lambda_t) - \xi. \tag{5.45}$$

From proposition 10 we know that $\|\nabla h(\lambda_t)\|$ converges to zero almost surely, therefore there exists a T such that $\|\bar{g}(\lambda_t)\| \leq \xi - \kappa$ for all $t \geq T$ for any $\kappa \in (0, \xi)$. For any particular (i,k), we have $\left|[\nabla h(\lambda_t)]_i^k\right| \leq \|\nabla h(\lambda_t)\|$, allowing us to conclude that

$$\mathbb{E}\left[q_i^k(t+1) \,\middle|\, \sigma_t\right] \leq q_i^k(t) - \kappa, \qquad \forall t \geq T \tag{5.46}$$

for all $q_i^k(t) \geq \kappa$ because $q_i^k(t) \geq 0$ for all t.

Recalling the definition of the stopped process in (5.39) it follows from (5.46) that there exists a time T such that

$$\mathbb{E}\left[\tilde{q}_i^k(t+1) \,\middle|\, \sigma_t\right] < \tilde{q}_i^k(t) \tag{5.47}$$

for all $\tilde{q}_i^k(t) > 0$, therefore $\tilde{q}_i^k(t)$ converges to zero almost surely. While $q_i^k(t)$ may move away from zero after $\tilde{q}_i^k(t)$ converges, we reinitialize the stopped martingale process to show that $q_i^k(t)$ must eventually return to zero. ■

Corollary 2. Consider the Stochastic Heavy Ball Routing Algorithm, defined by (5.29) with $g_t(\lambda_t)$ as defined in (5.14) and primal Lagrangian maximizers defined in (5.12). With step size sequence is chosen to satisfy $\sum_t \epsilon_t = \infty$ and $\sum_t \epsilon_t^2$, all queues become empty infinitely often with probability one,

$$\liminf_{t\to\infty} q_i^k(t) = 0, \quad \text{a.s.,} \qquad \text{for all } k, i \neq o_k. \tag{5.48}$$

Proof : From Proposition 10, we know that $||\nabla h(\lambda_t)||$ converges to zero which allows application of Proposition 11. From Proposition 11 it is guaranteed that each queue has a limit infimum equal to zero, which is equivalent to having each queue become empty infinitely often. ■

Corollary 2 is a sufficient condition for queue stability. In fact it is a much stronger result because the queues are always eventually cleared. This occurs because our dual variables converge to the optimal priorities which yield a routing $R_{ij}^k(\lambda_k)$ which forces the queue lengths to decrease in expectation whenever they are non-zero.

5.4 Numerical Results

The main benefit of introducing discounted integral priorities is the ability to drive large queues out of the system. The performance of the DIP routing in its simplest form, as defined in (5.25) is compared with the performance of the backpressure based queue stabilization methods defined in [Ribeiro, 2009b] and [Zargham et al., 2013a].

Our experiments take place on the network in Figure 4.3, with edge capacities chosen uniformly randomly in $(0, 50]$, the strictness inequality parameter $\xi = 0$ and there are $K = 5$ data types with unique destination nodes. Each queue starts with an expected initial backlog of 10 packets. The arrival process has an expectation of 5 packets per queue which results in an expected $5(n-1)K = 45$ packets entering the system at each time. The resulting problem (5.7) is feasible but non-trivial. Our choice of objective function is

$$f_{ij}^k(r_{ij}^k) = -\frac{1}{2}\left(r_{ij}^k\right)^2 + \beta_{ij}^k r_{ij}^k. \tag{5.49}$$

The quadratic term captures an increasing cost of routing larger quantities of packets across a link and help to eliminate myopic routing choices that lead to sending packets in cycles. The linear term

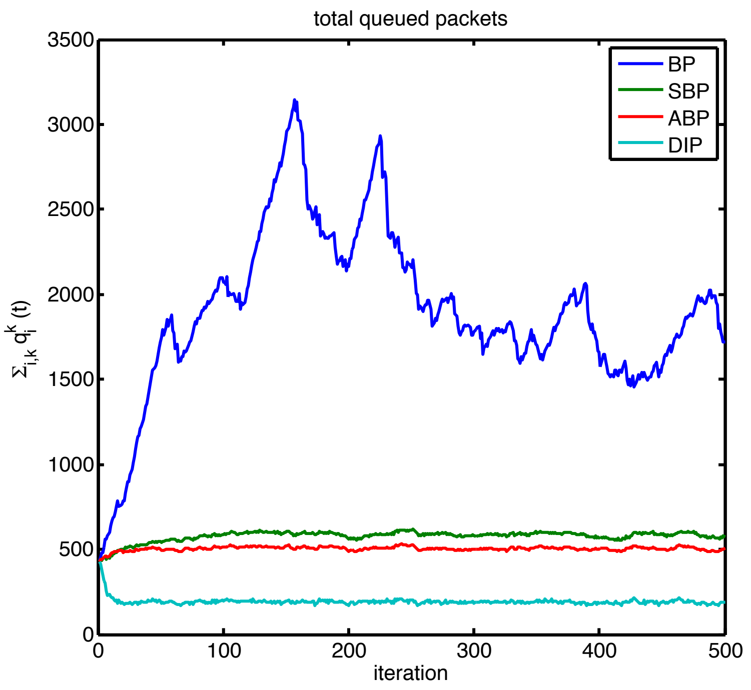


Figure 5.1: ABP and SBP stabilize the queues much more effectively than BP, but DIP routing drives the queues down and keeps them stable there.

β is introduced to reward sending packets to their destinations. In our simulations, $\beta_{ij}^k = 10$ for all edges routing to their respective data type destinations $j = o_k$ and all other i, j, k, $\beta_{ij}^k = 0$. Figure 5.1, is a characteristic example of the relative behaviors of the algorithms being studied. Backpressure (BP) stabilizes the queues but the total number of backlogged packets remains large. Accelerated Backpressure (ABP) stabilizes the queues more quickly that Soft Backpressure (SBP) but neither do any work to remove existing queue build up. Discounting Integral Priority (DIP) routing however drives the total packets queues significantly below their starting levels.

In order to more precisely characterize the relative behavior of these algorithms, we repeat the numerical experiment 100 times and record the *average-average* backlog. Two averages are used, we take a time average over 500 iterations to account for fluctuations and we average over the number of queues. Thus the data being presented is the average number of packets in each queue over the whole trial. Figure 5.2 is the distribution over all 100 trials. The key observation is that the Discounted Integral Priority (DIP) routing algorithm results in an average queue length of well

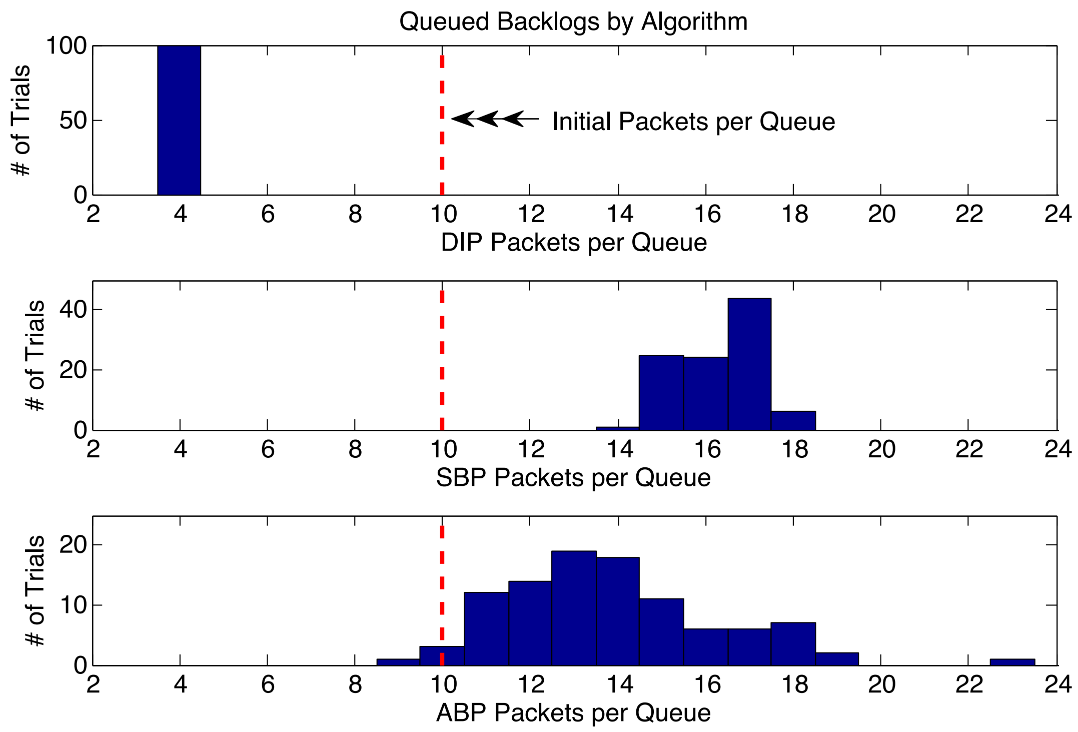


Figure 5.2: ABP and SBP stabilize the queues relatively quickly, preventing them from getting too far above the initial queues. ABP is faster than SBP but exhibits more variance. DIP routing however drives the queues well below the initial queue lengths.

below the initial average queue length while the other algorithms are at best close to the starting value. In fact the average backlog is close to the average arrival rate. Another important observation is highlighted in Table 5.1; these results are achieved with an extremely tight variance over many trials. Every realization of the problem has different edge capacities but the same arrival statistics. The implication is that the Discounted Integral Priorities successfully learn information about the arrival statistics and the remaining steady state backlogs result from the fact that the arrivals are stochastic.

Table 5.1: Average Queue Backlog Statistics

Algorithm	BP	SBP	ABP	DIP
Mean	74.13	16.22	13.92	4.38
St. Dev.	11.52	0.92	2.37	0.02

Chapter 6

Future Work

Having introduced a wide range of algorithms, it is important to ask where to go from here. In this chapter, we address the ways in which these algorthms can be generalized in order to improve performance or strengthen the analytical results. Since Accelerated Dual Descent (Chapter 3) is the most simplified version of the problem, it is the most natural place to generalize by weakening the assumptions and showing that our results still hold. In the cases of Accelerated Backpressure (Chapter 4) and Discounted Integral Priority Routing (Chapter 5), we observe a gap between the analytical results achieved for a decaying stepsize and the numerical experiments, which demonstrate performance for a fixed stepsize. Additionally, the Discounted Integral Priority Routing introduces a scheme with parallels to Nesterov type acceleration methods, implying that it may be applicable in a much broader class of problems.

6.1 Generalizations for Accelerated Dual Descent

The Accelerated Dual Descent family exploits the underlying structure of the convex network flow problem. In particular, computation of the Newton direction requires the inversion of a weighted graph Laplacian. This challenge is not unique to the network flow optimization problem; in fact, a whole branch of the consensus literature addresses problems of this form. A direction for continuing research on the Accelerated Dual Descent algorithm would be to apply it to other problems addressed in this literature.

A different, but related way to extend Accelerated Dual Descent, is to weaken the underlying assumptions. In the current formulation, strong assumptions are made on the primal problem in order to ensure that the dual problem satisfies the standard assumptions for the application of the Newton method. This set of assumptions limits the choice of primal objective to well conditioned, separable convex functions. The inclusion of multiple commodities and capacity constraints would actually allow for a broader class of objectives because one must only ensure the function satisfies these assumptions on a bounded domain.

Another important extension of the Accelerated Dual Descent research is to refine the results bounding the connectivity modulus. The connectivity modulus characterizes the information spreading rate in our problem. Thus, it fundamentally impacts the accuracy of our approximation of the Newton direction and consequently, the convergence rate results depend explicitly on this value. In Chapter 3, both combinatorial and spectral bounds on the connectivity modulus are presented, but neither bound is guaranteed to be particularly tight.

6.2 Variations of Accelerated Backpressure

The Accelerated Backpressure algorithm is derived using a matrix splitting similar to the one used in the Consensus Based Newton algorithm (Chapter 2). The Accelerated Dual Descent algorithm introduces an alternate splitting with an internal scaling factor which allows it to converge to the optimal point with a relatively large fixed stepsize. Further study of the Accelerate Backpressure algorithm should include a variation which makes use of the splitting introduced in Chapter 3.

The analytical results for Accelerated Backpressure require a decaying stepsize in order to guarantee queue stability. In light of a more complete characterization of the internal scaling effect in Accelerated Dual Descent, it may be possible to prove that Accelerated Backpressure ensures queue stability for a relatively large fixed stepsize. Such a result would be consistent with the results we observe in our numerical experiments.

6.3 Applications for Discounted Integral Priorities

The Discounted Integral Priority Routing is the application of a more general optimization tool to the specific case of the Priority Based Routing. The fundamental concept is the use of a discounted integrator to smooth the effects of stochastic gradient descent. The next step for this approach is to analyze it for a more general optimization problem. Structurally, discounted integral priorities are closer to momentum methods like those introduced by Polyak and Nesterov. Preliminary numerical experiments show that it can be used to accelerate more sophisticated dual descent methods such as Accelerated Dual Descent and Alternating Direction Method of Multipliers (ADMM).

Our analytical results are defined for a decaying stepsize; as in the case for Accelerated Back-pressure, the Discounted Integral Priority Routing requires the decaying stepsize to ensure queue stability. In numerical experiments, Discounted Integral Priority Routing not only stabilizes the queues, but visibly reduces the queues at steady state. It is probable that a queue stability result that does not require a decaying stepsize exists. The best approach for finding such a proof would be to explore the relationship between discounted integral priorities and dual Stochastic Average Gradient (SAG) and dual Stochastic Gradient Descent (SGD). With only minor adjustments to the algorithm, including selection of the appropriate time discounting factor, one can recover both of these well studied stochastic optimization methods from discounted integral dual descent.

Bibliography

R. Adhikari, N. Aste, and M. Manfren. Multi-commodity network flow models for dynamic energy management – smart grid applications. *Energy Procedia*, 14:1374 – 1379, 2012.

J. Al-Karaki and A. Kamal. Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, 11(6):6–28, Dec 2004.

D. Aldous. Some inequalities for reversible Markov chains. *Journal of the London Mathematical Society*, 25:564–576, 1982.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009a.

A. Beck and M. Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *Image Processing, IEEE Transactions on*, 18(11):2419–2434, 2009b.

A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. Academic Press, New York, 1979.

Bertsekas and Gafni. Projected newton methods and optimization of multi-commodity flow. *IEEE Transactions on Automatic Control*, 28:1090–1096, 1983.

D. Bertsekas. *Nonlinear Programming*. Athena Scientific, Cambridge, Massachusetts, 1999.

D. Bertsekas, A. Nedić, and A. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, Cambridge, Massachusetts, 2003.

D. P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, Belmont, MA, 1998.

D. P. Bertsekas and D. E. Baz. Distributed asynchronous relaxation methods for convex network flow problems. *SIAM Journal of Optimization and Control*, 25:74–85, 1987.

J. F. Bonnans, J. C. Gilbert, C. Lemarechal, and C. A. Sagastizábal. *Numerical optimization: Theoretical and practical aspects*. Springer-Verlag, Berlin, Germany, 2006.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

A. Chen and A. Ozdaglar. A fast distributed proximal-gradient method. *Proc. of Allerton Conference on Communication, Control, and Computing*, 2012.

S. Chen and R. Saigal. A primal algorithm for solving a capacitated network flow problem with additional linear constraints. *Networks*, 7(1):59–79, 1977.

M. Chiang, S. Low, A. Calderbank, and J. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, 2007.

F. Chung. *Spectral Graph Theory*. The American Mathematical Society, 1997.

R. Dembo, S. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19:400–408, 1982.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. In *INFOCOM 2005. 24th Annual Joint Conference of the*

IEEE Computer and Communications Societies. Proceedings IEEE, volume 3, pages 1794–1803. IEEE, 2005.

A. Eryilmaz and R. Srikant. Joint congestion control, routing, and mac for stability and fairness in wireless networks. *Selected Areas in Communications, IEEE Journal on*, 24(8):1514–1524, Aug 2006. ISSN 0733-8716. doi:10.1109/JSAC.2006.879361.

J. Fink, A. Ribeiro, and V. Kumar. Robust control for mobility and wireless communication in cyber–physical systems with application to robot teams. *Proceedings of the IEEE*, 100(1):164–178, 2012.

S. D. Flam. Optimization under uncertainty using momentum. *Lecture Notes in Economics and Mathematical Systems*, pages 249–256, 2004.

N. Gatsis, A. Ribeiro, and G. B. Giannakis. A class of convergent algorithms for resource allocation in wireless fading networks. *IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS*, 9, 2010.

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

A. V. Goldberg, E. Tardos, and R. E. Tarjan. *Network Flow Algorithms*. Springer-Verlag, Berlin, 1990.

A. Guerraggio, D. Luc, and N. Minh. Second-order optimality conditions for c1 multiobjective programming problems. *ACTA Mathematica Vietnamica*, 26(3):257–268, 2001.

Y. T. Herer and M. Tzur. The dynamic transshipment problem. *Naval Research Logistics (NRL)*, 48 (5):386–408, 2001.

H. Hindi and W. Ruml. Network flow modeling for flexible manufacturing systems with re-entrant lines. In *Decision and Control, 2006 45th IEEE Conference on*, pages 1043–1050, Dec 2006.

A. Hoffman and S. R. Moore graphs with diameter 2 and 3. *IBM Journal of Research and Development*, 5(4):497–504, 1960.

R. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, New York, 1985.

A. Jadbabaie, J. Lin, and S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

A. Jadbabaie, N. Motee, and M. Barahona. On the stability of Kuramoto model of coupled nonlinear oscillators. In *Proceedings of the American Control Conference*, June 2004.

A. Jadbabaie, A. Ozdaglar, and M. Zargham. A distributed newton method for network optimization. In *Proceedings of IEEE CDC*, 2009.

C. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, Philadelphia, PA, 1995.

C. T. Kelley. *Solving Nonlinear Equations with Newton's Method*. SIAM Fundamentals of Algorithms, 2003.

F. Kelly, A. Maulloo, and D. Tan. Rate control for communication networks: shadow prices, proportional fairness, and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

J. G. Klincewicz. A newton method for convex separable network flow problems. *Bell Laboratories*, 1983.

V. Kolmogorov and A. Shioura. New algorithms for the dual of the convex cost network flow problem with application to computer vision. *Tech Report, Mathematical Programming*, 2007.

H. Landau and A. Odlyzko. Bounds for eigenvalues of certain stochastic matrices. *Linear Algebra and its Applications*, 38:5–15, 1981.

L. Lessard, B. Recht, and A. Packard. Analysis and design of optimization algorithms via integral quadratic constraints. private communications, 2014.

Q. Ling and A. Ribeiro. Decentralized dynamic optimization through the alternating direction method of multipliers. In *Proc. IEEE Workshop on Signal Process. Advances in Wireless Commun.*, 2013.

I. Lobel and A. Ozdaglar. Distributed subgradient methods for convex optimization over random networks,. *IEEE Transactions on Automatic Control*, 56, 2011.

S. Low and D. Lapsley. Optimization flow control, I: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, 1999.

B. Mohar and Y. Alavi. The laplacian spectrum of graphs. *Graph Theory, Combinatorics and Applications*, 2:871–898, 1991.

M. R. Murty. Ramanujan graphs. http://www.mast.queensu.ca/ murty/ramanujan.pdf; Department of Mathematics, Queens University, Kingston, Ontario, January 2003.

A. Nedić and A. Ozdaglar. Subgradient methods in network resource allocation: Rate analysis. In *Proc. of CISS*, 2008.

A. Nedić and A. Ozdaglar. Approximate primal solutions and rate analysis for dual subgradient methods. *SIAM Journal on Optimization*, 19:1757–1780, 2009a.

A. Nedić and A. Ozdaglar. Distributed subradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009b.

M. J. Neely. Energy optimal control for time varying wireless networks. *IEEE Transactions on Information Theory*, 52:2915–2934, 2006.

M. J. Neely, E. Modiano, and C. E. Rohrs. Dynamic power allocation and routing for time varying wireless networks. *IEEE Journal on Selected Areas in Communications, Special Issue on Wireless Ad-Hoc Networks*, 23:89–103, 2005.

M. J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. *IEEE Transactions on Networking*, 16:396–409, 2008.

Y. Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

M. E. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

R. Olfati-Saber. Ultrafast consensus in small-world networks. In *Proceedings of IEEE ACC*, 2005.

R. Olfati-Saber and R. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

A. Olshevsky and J. Tsitsiklis. Convergence speed in distributed consensus and averaging. *SIAM Journal on Optimization*, 2006.

J. B. Orlin. Minimum convex cost dynamic network flows. *MATHEMATICS OF OPERATIONS RESEARCH*, 9(2):190–207, May 1984.

R. Peng and D. Spielman. An efficient parallel solver for sdd linear systems. In *Symposium on the Theory of Computing*, 2014.

B. Polyak. A general method for solving extremum problems. *Soviet Math. Doklady*, 8(3):593–597, 1967.

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

A. Ribeiro. Ergodic stochastic optimization algorithms for wireless communication and networking. *IEEE Transactions on Signal Processing*, 2009a.

A. Ribeiro. Stochastic soft backpressure algorithms for routing and scheduling in wireless ad-hoc networks. In *IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, 2009b.

A. Ribeiro and G. Giannakis. Separation principles in wireless networking. *IEEE Transactions on Information Theory*, 58:4488–4505, 2010.

A. Ribeiro and G. B. Giannakis. Separation theorems of wireless networking. *IEEE Transactions on Information Theory*, 2007.

R. T. Rockafellar. *Network Flows and Monotropic Programming*. Wiley, New York, NY, 1984.

Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.

M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *http://arxiv.org/abs/1309.2388*, 2013.

N. Z. Shor. *Minimization Methods for Non-differentiable Functions*. Springer, 1985.

Shu-Cherng and L. Qi. Manufacturing network flows: A generalized network flow model for manufacturing process modelling. *Optimization Methods and Software*, 18(2):143–165, April 2003.

A. J. Sinclair. Improved bounds for mixing rates of Markov chains and multicommodity flow. *Combinatorics, Probability and Computing*, 1:351–370, 1992.

V. Solo and X. Kong. *Adaptive Signal Processing Algorithms: Stability and Performance*. Prentice-Hall, 1995.

D. Spielman. Algorithms, graph theory, and the solution of laplacian linear equations. In A. Czumaj, K. Mehlhorn, A. Pitts, and R. Wattenhofer, editors, *Automata, Languages, and Programming*, volume 7392 of *Lecture Notes in Computer Science*, pages 24–26. Springer Berlin Heidelberg, 2012.

J. Sun and H. Kuo. Applying a newton method to strictly convex separable network quadratic programs. *SIAM Journal of Optimization*, 8:728–745, 1998.

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

P. Tseng. An incremental gradient(-projection) method with momentum term and adaptive stepsize rule. *SIAM Journal on Optimization*, 8(2):506–531, 1998.

P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *self-published: http://pages.cs.wisc.edu/ brecht/cs726docs/Tseng.APG.pdf*, 2008.

K. I. Tsianos, S. Lawlor, and M. G. Rabbat. Communication/computation tradeoffs in consensus-based distributed optimization. In *NIPS*, pages 1952–1960, 2012.

R. Tutunov, M. Zargham, and A. Jadbabaie. On convergence rate of accelerated dual descent algorithm. In *Proceedings of IEEE CDC*, 2014.

D. Watkins. *The Matrix Eigenvalue Problem*. SIAM, 2007.

D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

E. Wei and A. Ozdaglar. Distributed alternating direction method of multipliers. In *Proceedings of IEEE CDC*, 2013.

Y. Wu, A. Ribeiro, and G. B. Giannakis. Robust routing in wireless multi-hop networks. *Proc. Conf. on Info. Sciences and Systems*, 2007.

M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie. Accelerated dual descent for network optimization. In *Proceedings of IEEE ACC*, 2011.

M. Zargham, A. Ribeiro, and A. Jadbabaie. Network optimization under uncertainty. *Proceedings of IEEE CDC*, 2012.

M. Zargham, A. Ribeiro, , and A. Jadbabaie. Accelerated backpressure algorithm. *Proceedings of the IEEE GLOBECOM*, 2013a.

M. Zargham, A. Ribeiro, and A. Jadbabaie. Accelerated dual descent for constrained convex network flow optimization. *Proceedings of IEEE CDC*, 2013b.

M. Zargham, A. Ribeiro, and A. Jadbabaie. Discounted integral priority routing for data networks. In *Proc. of the Global Communications Conference*, December 2014a.

M. Zargham, A. Ribeiro, and A. Jadbabaie. Accelerated backpressure algorithm. *IEEE Transactions on Signal Processing (submitted)*, 2014b.

M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie. Accelerated dual descent for network flow optimization. *IEEE Transactions on Automatic Control*, 59(4):905–920, April 2014c.