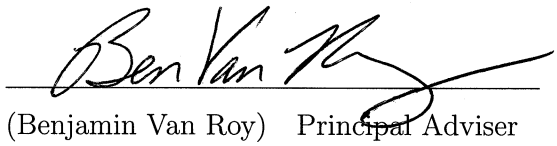
A MESSAGE-PASSING PARADIGM FOR OPTIMIZATION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

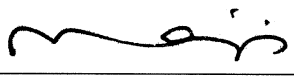
Ciamac Cyrus Moallemi
September 2007

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.



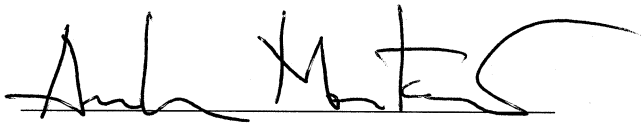
(Benjamin Van Roy) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.



(Balaji Prabhakar)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.



(Andrea Montanari)

Approved for the University Committee on Graduate Studies.

ABSTRACT

We consider a class of large-scale optimization programs that appear in many engineering and management settings. Such complex, decentralized systems are comprised of many individual components, each interacting with a limited number of neighboring components. The resulting optimization programs are characterized by the fact that the decision variables are coupled only in a localized and sparse fashion.

Message-passing algorithms are a new class of distributed and asynchronous methods for solving problems with graphical structure. They have emerged independently in a number of fields, and interest in these algorithms has been motivated by their empirical success as an approximation method for certain intractable problems. However, they are still poorly understood in the general optimization context. The objective of this thesis is to understand these algorithms and elucidate their relationship to standard analytical and algorithmic optimization techniques.

We first consider message-passing in the context of resource allocation problems. We demonstrate that message-passing provides a framework for decentralized management that generalizes classical price-based systems by allowing incentives to vary across activities and consumption levels. We demonstrate that message-based incentives lead to system-optimal behavior for convex resource allocation problems, yet yield allocations superior to those from price-based incentives for nonconvex resource allocation problems. We describe and demonstrate the resulting distributed and asynchronous protocol in the context of a network resource allocation problem.

Next, we consider the application of message-passing algorithms to the solution of unconstrained, convex optimization programs. We establish that message-passing asynchronously converges for a large class of such programs that are characterized by a scaled diagonal dominance condition. This condition is similar to known sufficient conditions for asynchronous convergence of other decentralized optimization algorithms, such as coordinate descent and gradient descent.

Finally, we apply message-passing to the distributed consensus problem, where a collection of numbers must be averaged across a network in a distributed and asynchronous fashion. We demonstrate that the resulting protocol, consensus propagation, converges, characterize the convergence rate for regular graphs, and show that the protocol exhibits better scaling properties than methods based on linear consensus, which is an alternative that has received much recent attention.

For Ben Van Roy
il miglior fabbro

ACKNOWLEDGMENTS

The research in this thesis is the result of collaborative work with my advisor, Professor Benjamin Van Roy. He has taught me a great many things, but from him I have principally learned the virtue of clarity in thought and deed. I am deeply grateful for his advice and his friendship.

I would like to particularly thank Professor Balaji Prabhakar, who served as a thesis reader, and more importantly, as a intellectual mentor. He has been unfailingly generous with his erudition on all aspects of life.

I am further indebted to my thesis reader Professor Andrea Montanari, and the other members of my oral examination committee, Professors Ramesh Johari and Sunil Kumar, both for their efforts on my behalf, and as I have profited much from our interactions. My discussions with Professors Martin Wainwright, Devavrat Shah, and Stephen Boyd have shaped the work in this thesis. Professor Tsachy Weissman has been a thoughtful teacher and unselfish collaborator.

Vivek Farias has been my officemate and collaborator during my entire time at Stanford. The path of shared experience we leave behind has forged a friendship which I look forward to on the road ahead.

I have greatly benefited from many discussions and interactions with fellow students at Stanford. I especially wish to acknowledge Pieter Abbeel, Chandra Nair, Mohsen Bayati, and James Mammen, among others.

I am grateful to Joëlle Skaf for proofreading this thesis, which is only one of the many examples of patience, kindness, and companionship that she has offered me during the past year.

Finally, I would like to thank my parents, Karim Moallemi and Azar Tadayyoni, for their love, support, and encouragement has been invaluable.

The author of this work was supported by a Benchmark Stanford Graduate Fellowship. This work was further supported, in part, by the National Science Foundation under Grant IIS-0428868 and a supplement to Grant ECS-9985229 provided by the Management of Knowledge Intensive Dynamic Systems Program (MKIDS).

PRELIMINARIES

Notation

Throughout this thesis, equality by definition is written as $A \triangleq B$, and serves to define A by B.

We use $\mathbb{R}$ to denote the real numbers, $\mathbb{R}_+$ to denote the nonnegative real numbers $[0, +\infty)$, and $\mathbb{R}_{++}$ to denote the positive real numbers $(0, +\infty)$. Given a finite set V and a set $\mathcal{X}$, we use $\mathcal{X}^V$ to denote the product space of vectors whose components are indexed by the set V, and each lie in the space $\mathcal{X}$. Given a vector $x \in \mathcal{X}^V$ and a subset of component indices $C \subset V$, we denote by $x_C \in \mathcal{X}^C$ the vector consisting of the subset of components of x indexed by the set C. That is, $x_C \triangleq (x_i \ : \ i \in C)$. We denote by $\mathbf{1} \triangleq (1, \ldots, 1) \in \mathbb{R}^n$ a vector where every component is 1.

Definitions and notation from graph theory will play an important role in this thesis. An *undirected graph* (V, E) consists of a set of vertices V, and a set of edges E, where each edge is an unordered pair of distinct vertices. For a vertex $i \in E$, we denote by $N(i) \triangleq \{j \in V : \ (i, j) \in E\}$ the set of neighboring vertices. In some instances it will be necessary to distinguish direction on edges, so we define the set $\vec{E} \triangleq \{(i, j) \in V \times V : \ i \in N(j)\}$. If $i \in V$ and $j \in N(i)$ are neighboring vertices, we allow (i, j) to refer to both an unordered element of E and an ordered element of $\vec{E}$, depending on the context.

A *hypergraph* $(V, \mathcal{C})$ consists of a finite set of vertices V, and a collection $\mathcal{C}$ of hyperedges. Each hyperedge $C \in \mathcal{C}$ is a nonempty subset of the vertex set C, i.e., $C \subset V$. We denote by $\partial i \triangleq \{C \in \mathcal{C} : \ i \in C\}$ the set of hyperedges incident to a vertex $i \in V$.

Bibliographic Notes

Much of the original research in this thesis has has been published previously. The material in Chapter 3 is based on work in [62]. The material in Chapter 4 is based on work in [60, 61]. The material in Chapter 5 is based on work in [58, 59].

CONTENTS

LIST OF FIGURES

1

INTRODUCTION

Over the past century, optimization has become a powerful tool in the design, analysis, and implementation of engineering and management systems. The mathematical programming community has had a broad and important impact through the development of an "optimization toolbox" which seeks to exploit common structures that appear in optimization problems in many applied settings. Both theoretical and computational methods have been developed for linear, convex, integer, and dynamic programming, for example, which allow practitioners to focus on application-specific modeling issues rather than optimization issues.

In this thesis, we consider a different type of structure that commonly appears in large-scale optimization problems. In many such cases, although there may be a large number of decision variables, these variables are coupled in a rather limited way. A complex objective function or set of constraints may decompose so that each term contains only a small number of decision variables. Such localized, sparse interactions can be described using the metaphor of a graph.

Many problems with graphical structure arise in real-world applications in which a system consists of many distinct components. Each component is only responsible for a small subset of the decision-making process, and components naturally interact only with neighboring components that are, for example, geographically proximate. In such systems, finding solutions in a distributed fashion, in the absence of a central coordinating authority, are of particular importance.

Message-passing algorithms are a new class of decentralized and asynchronous methods for solving problems with graphical structure. They have emerged independently in a number of fields: communications theory, artificial intelligence, and statistical physics. Interest in these algorithms has been motivated by their empirical success as an approximation method for certain classes of intractable problems, such as those arising in the decoding of error-correcting codes, or certain Boolean satisfiability problems. A body of theoretical work is emerging, but these methods are still poorly understood in the general optimization context. Moreover, they have received little attention in the mathematical programming community. Our objective in this thesis is to understand the properties of these algorithms and elucidate their relationship to standard analytical and algorithmic optimization techniques. We hope that message-passing algorithms might become a new and valuable addition to the optimization toolbox.

1.1 Graphical Models

Many optimization problems involve only localized interactions between decision variables. These interactions can be encoded as a graph as follows: consider an optimization problem associated with a hypergraph $(V, \mathcal{C})$. There are $|V|$ decision variables; each is associated with a vertex $i \in V$ and takes values in a set $\mathcal{X}$. The set $\mathcal{C}$ is a collection of subsets (or, "hyperedges") of the vertex set V; each hyperedge $C \in \mathcal{C}$ is associated with an extended real-valued function (or, "factor") $f_C : \mathcal{X}^C \to \mathbb{R} \cup \{+\infty\}$. The optimization problem takes the form

$$\begin{array}{lll} \underset{x}{\text{minimize}} & F(x) \triangleq & \sum_{C \in \mathcal{C}} f_C(x_C) \\ \text{subject to} & & x_i \in \mathcal{X}, \qquad \forall\, i \in V. \end{array} \tag{1.1}$$

Here,

$$x_C \triangleq (x_i : \ i \in C)$$

is the collection of variables associated with the vertices of the hyperedge C.

We refer to programs of the form (1.1) as *graphical models*. Here, the relationships between decision variables within the terms of the additive decomposition are highlighted. By allowing for extended real-valued terms in the objective function, this formulation also supports constrained optimization problems.

Note that almost any optimization program can be trivially represented in the form (1.1), by considering a single factor that involves all of the decision variables. However, implicit in this formulation is the assumption that the underlying hypergraph is sparse, in the sense that each factor involves a small number of decision variables.

1.2 Motivating Applications

While the graphical model formulation is quite general, it is instructive to consider the following applications. They will provide motivation for the methods presented in this thesis.

1.2.1 Network Rate Allocation

Consider a communications network consisting of a set of users $\mathcal{A}$ and a set of communications links $\mathcal{R}$. Each user $a \in \mathcal{A}$ wishes to transmit data across a particular route in the network. If the user a is allocated transmission capacity $x_a \in \mathbb{R}_+$, the user derives utility $u_a(x_a)$. Each link $r \in \mathcal{R}$ has a finite capacity $b_r > 0$. This capacity must be shared by the set of users which require along the link, denote this set by $\partial r \subset \mathcal{A}$. The network manager's optimization problem is to allocate capacity so as to maximize social welfare; that is, according to the solution of the optimization problem

$$\begin{array}{lll} \underset{x}{\text{maximize}} & \sum_{a\in\mathcal{A}} u_a(x_a) & \\ \text{subject to} & \sum_{a\in\partial r} x_a \leq b_r, & \forall\ r \in \mathcal{R}, \\ & x_a \geq 0, & \forall\ a \in \mathcal{A}. \end{array} \tag{1.2}$$

Such problems typically possess a sparse graphical structure. While the network may be enormous, with many users and links, each user will require service along only a small subset of the links, and each link will provide service to a small subset of the users.

Interest in this network control example was largely spurred by the work of Kelly [40] in the analysis of rate control protocols. Subsequently, a large literature has emerged that examines a broad class of network control problems using optimization-based methods (see [23] for a survey). The spirit here is that the control of a network should be viewed as the solution of a global optimization problem, with decision variables that correspond to the available controls (e.g., transmission rates, power levels, scheduling and routing decisions), constraint sets that incorporate the physical limitations of the network (e.g., capacity constraints, interference constraints), and objective functions that capture end-user utility (e.g., throughput, delay). New control protocols can be designed as algorithms to solve the underlying global optimization problem, while existing protocols can be reverse-engineered to learn the utility functions they implicitly seek to optimize.

The numbers of users and links in modern communication networks are enormous. As such, it is not possible for a central authority to gather all the information (e.g., utility functions, link capacities) that would be required to make centralized control decisions. Rather, these decisions must be made in a distributed manner, based on locally available information, with communications that occur alongside the normal flow of network traffic.

1.2.2 Distributed Estimation

Sensor networks are a modern architecture for performing environmental sensing tasks [2]. A sensor network consists of a collection of low-cost elements, each with some limited sensing, computation, and communication capabilities. These sensors are distributed within an environment in order to collectively perform a monitoring or estimation task. We describe a prototypical such estimation task as follows:

Markov random fields [44] are a common framework for describing the spatially distributed random phenomena that are monitored by sensor networks. In such a setting, there is a vector of random variables $x \in \mathcal{X}^V$ that are to be estimated, indexed by a set V, where each random variable x_i takes values in a set $\mathcal{X}$. The vector x is distributed according to the distribution

$$P(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C).$$

Here, $\mathcal{C}$ is a collection of subsets of V, each function $\psi_C : \mathcal{X}^C \to \mathbb{R}_{++}$ is called a potential function, and the scalar $Z > 0$ is a normalization constant.

There is a sensor associated with each component x_i, and this sensor makes a noisy observation $y_i \in \mathcal{X}$ of the value of x_i. It is assumed that, conditioned on x, each observation y_i is independently distributed according to the distribution $p_i(\cdot|x_i)$. The estimation task is to compute a maximum *a posteriori* estimate of the vector x. This can be accomplished by solving the optimization problem

$$\underset{x \in \mathcal{X}^V}{\text{maximize}} \ \sum_{i \in V} \log p_i(y_i|x_i) + \sum_{C \in \mathcal{C}} \log \psi_C(x_C). \tag{1.3}$$

Sensor elements, however, have a limited communication ability. They may be able to communicate wirelessly with other sensors in a short range, but such communication must be parsimonious due to the limited energy supply available to each sensor. The topology of the network may be dynamic and changing, due to the environment. Centralized estimation schemes, which require all observations to be transmitted to a base station, may require too much communication and coordination. Instead, decentralized estimation methods are required. Here, sensor elements communicate amongst each other to perform data fusion, so as to economize communications and to distribute information processing tasks.

1.2.3 Distributed Consensus

Consider a network of processors whose topology is described by an undirected, connected graph (V, E). Each processor i is given a value $y_i \in \mathbb{R}$. The processors

wish to collectively compute the average

$$\bar{y} \triangleq \frac{1}{|V|} \sum_{i \in V} y_i.$$

However, they are restricted to communicating only along the edges in E. The *distributed consensus* problem is to compute the average $\bar{y}$ in a decentralized, asynchronous way.

Distributed consensus can be formulated as a an optimization problem by associating a decision variable $x_i \in \mathbb{R}$ with each processor $i \in V$, and solving

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \sum_{i \in V} (x_i - y_i)^2 \\ \text{subject to} & x_i = x_j, \qquad \forall\ (i,j) \in E. \end{array} \tag{1.4}$$

Since the graph is assumed to be connected, it is easy to verify that the unique solution x^* to this program satisfies $x_i^* = \bar{y}$, for all $i \in V$. Moreover, the graphical structure of the program (1.4) corresponds exactly to the original graph (V, E).

Distributed consensus has appeared as an important problem in a number of different settings in the past few decades. It is one of the first and most basic problems in distributed computation [87, 14]. It can be viewed as a maximum likelihood estimation problem, where each processor makes an observation of a common value that is corrupted by independent and identically distributed Gaussian noise. Hence, it is important for data fusion or aggregation in sensor networks. There are, however, a broad array of other applications, such as load balancing [26, 68, 28], clock synchronization [48, 92], and coordinated control of autonomous agents [38, 49, 69, 66, 72, 82].

1.3 Classical Decentralized Algorithms

A common theme among the applications in Section 1.2 is that, in many graphical models, the graphical structure emerges from real physical constraints in the underlying problem. Vertices in the graphical model (decision variables) often correspond to distinct physical entities. Each such entity is limited to communicating

with neighboring entities to which it is naturally coupled. These neighboring entities are precisely those connected across hyperedges of the graph.

In such applications, distributed and asynchronous algorithms are crucially important for a number of reasons. The informational requirements of centralized solutions are often prohibitive. They may entail significant communications, and may require additional synchronization overhead. Moreover, even in instances where communication is not a binding constraint, computational capacity may well be limited. Decentralized solutions distribute the information process burden across many processors, and may allow for the solution of much larger scale problems.

Traditionally, a class of algorithms, which we will call *local search algorithms*, have been applied to obtain solutions in a decentralized and asynchronous manner [14]. It is illustrative to discuss two prototypical local search algorithms, *coordinate descent* and *gradient descent*.

Coordinate descent is an iterative procedure that seeks to solve the optimization problem (1.1). At each integer time t, denote by $x^{(t)}$ the current estimate of the optimal solution. Each processor i is responsible for the ith component $x_i^{(t)}$. This component is updated by seeking the value of x_i that minimizes the global objective function $F(\cdot)$, assuming that all of the other decision variables are held fixed at their values at time t. In other words,

$$x_i^{(t+1)} \triangleq \underset{x_i}{\operatorname{argmin}} \; F\left(x_{V\setminus i}^{(t)}, x_i\right).$$

Examining the objective function $F(x)$, it is clear that this is equivalent to

$$x_i^{(t+1)} = \underset{x_i}{\operatorname{argmin}} \sum_{C\in\partial i} f_C\left(x_{C\setminus i}^{(t)}, x_i\right),$$

where $\partial i \triangleq \{C \in \mathcal{C} : \; i \in C\}$ is the set of hyperedges incident to vertex i. Hence, in order to update the component x_i, processor i needs only knowledge of the current estimates of components of processors neighboring i in the hypergraph.

In the case where the domain $\mathcal{X}$ for each decision variable is continuous and the objective function is differentiable, one might also consider the *gradient descent* update

$$\begin{aligned} x_i^{(t+1)} &\triangleq x_i^{(t)} - \lambda \frac{\partial}{\partial x_i} F\left(x^{(t)}\right) \\ &= x_i^{(t)} - \lambda \sum_{C \in \partial i} \frac{\partial}{\partial x_i} f_C\left(x_C^{(t)}\right). \end{aligned}$$

Here, the scalar λ is a step-size parameter. As with coordinate descent, this update can be performed by a processor with only local knowledge of neighboring estimates.

While the algorithms described above involve parallel and synchronous updates, it is easy to imagine asynchronous variations. In such schemes, each processor would keep track of the coordinate values for each of its neighbors in the hypergraph. Processors would occasionally update their coordinate values, and occasionally communicate these values to their neighbors. Communication occurs only over edges in the hypergraph, and processors need only be aware of their local neighborhood. In particular, no global knowledge or coordination of the entire system is needed.

Local search algorithms yield decentralized and asynchronous algorithms that work well in many problem instances. However, they have two significant problems:

Poor quality solutions. In general, there is no guarantee that local search algorithms will yield optimal solutions, or even converge at all. A solution which is a fixed point of coordinate descent can only be guaranteed to be optimal relative to the set of feasible points which can be obtained by altering only any single component. Significant improvement of the objective value may be possible, however, by simultaneously perturbing multiple components of the solution.

Similarly, for gradient descent, a fixed point is only guaranteed to be locally optimal, so that it cannot be improved by infinitesimal deviations. Without additional assumptions, such as convexity, there may be large deviations to the solution vector that result in better solutions.

Slow rate of convergence. For some classes of problems, convergence to optimal solutions can be guaranteed. One such example is when gradient descent is applied to an optimization problem that is convex. In this case, under appropriate technical assumptions, gradient descent will converge and yield a globally optimal solution. Unfortunately, gradient descent can be notoriously slow to converge. Higher order techniques, such as Newton's method, address this. However, these methods are fundamentally centralized.

The message-passing algorithms we discuss in the next section seek to address these deficiencies.

1.4 Message-Passing Algorithms

Over the past few years, there has been an explosion of interest in a new class of algorithms for problems with graphical structure. These methods, which we collectively refer to as *message-passing algorithms*, have been employed to solve probabilistic inference and optimization problems, and also as analytical tools in the study of random ensembles of systems. They have been independently discovered in a number of different fields. In communications theory, such methods were developed by Gallager [35] as algorithms for the decoding of error-correcting codes. In artificial intelligence, the loopy variation of the belief propagation algorithm of Pearl [70] is a message-passing algorithm for solving probabilistic inference problems. In statistical physics, the cavity method of Mézard, Parisi, and Virasoro [55] is a message-passing approach for the analysis of models in condensed matter physics.

We consider message-passing algorithms as methods for solving optimization problems of the form (1.1). Algorithms of this type are known in the literature under names such as belief revision or the max-product or min-sum algorithms [70, 1]. These operate by imagining a processor associated with each vertex and hyperedge in the graphical model. Processors exchange "messages" with their neighbors, according to the topology of the graph. These messages are updated

iteratively as the algorithm proceeds, and are used to compute estimated optimal values of each decision variable.

Message-passing algorithms are extremely general in that their precise form is simple and follows immediately from the particular choice of graphical decomposition (1.1). No problem specific analysis is required. Further, because of their localized nature, they can be implemented in a completely decentralized and asynchronous fashion.

Interest in message-passing algorithms has largely been triggered by the success of "turbo decoding" [11, 10, 34, 74]. Turbo decoding is now used routinely in communication systems that employ error-correcting codes. The decoding problem it aims to solve is NP-hard, and it was a surprise that this simple and efficient algorithm offers excellent approximate solutions.

Separately, inspired by ideas from statistical physics, message-passing algorithms have been proposed for solving certain NP-hard combinatorial optimization problems such as satisfiability and graph coloring [56, 21, 6, 22, 53]. While one cannot hope for efficient algorithms for these problems in the general case, this work, called "survey propagation," has demonstrated remarkable success over random ensembles of problems.

There has been much work dedicated to understanding whether these algorithms can be useful in other contexts. A body of empirical work in various problem domains demonstrates promise in areas such as data mining (e.g., [54, 71]), computer vision (e.g., [31, 25, 83]), and bioinformatics (e.g., [46, 47, 84]).

One may be tempted to dismiss message-passing algorithms as yet another ad hoc, heuristic approach to optimization. However, in some of the instances noted above — in particular, decoding and satisfiability — message-passing algorithms represent the state-of-the-art method of solution. Moreover, message-passing algorithms bear rich structure that can be employed as an analytical tool. For example, in the coding context, it has been shown that using certain coding schemes in conjunction with message-passing algorithms to solve NP-hard problems at the decoder offers near optimal average communication performance [73]. Further, used

as a conceptual rather than a computational tool, message-passing algorithms have led to analytic solution of large scale combinatorial optimization problems [3, 85].

But the theory of graphical models and message-passing algorithms is far from settled. Existing results are somewhat disparate, often customized to particular applied contexts or requiring varied technical assumptions. Though there is some intuition, to some extent guided by folklore propagated in the associated scientific community, it is generally unclear how well message-passing algorithms should work for any given application or what factors influence this. Experimental studies and analyses are carried out on a case by case basis. There is a pressing need for a coherent theory.

1.5 Advantages of Message-Passing Algorithms

Despite the broad interest in message-passing algorithms and their great empirical successes, these methods are largely unknown within the operations research community. In this thesis, we seek to better understand the properties of message-passing algorithms in the context of decentralized optimization. In the chapters that follow, our central theme will be to argue that message-passing algorithms can address the shortcomings of the conventional decentralized methods discussed in Section 1.3, namely:

Higher quality solutions. In general, the fixed points of message-passing algorithms will not yield globally optimal solutions. This is to be expected, as many of the problems under consideration are NP-hard. Any method that guarantees optimality is unlikely to be of practical use.

However, message-passing algorithms will yield solutions with stronger optimality guarantees than those provided by coordinate descent or gradient descent. Furthermore, in practice, the improvement in solution quality is significant.

Faster convergence. Message-passing algorithms can offer a better rate of convergence than conventional decentralized methods. This is because the messages

convey richer information than the coordinate estimates that are exchanged by local search methods. Further, flow of information in a message-passing algorithm is somehow directed, as opposed to the diffusive flow of information in a local search algorithm.

This thesis also offers important contributions to the understanding of message-passing algorithms. In particular, we are able to offer a new interpretation of message-passing fixed points as generalizations of classical ideas in convex analysis. We are able to establish convergence for a broad class of convex optimization problems, and, in a particular instance of interest, even results on the rate of convergence.

1.6 Organization of This Thesis

The balance of this thesis is organized as follows:

Chapter 2. In this chapter, we provide a self-contained introduction to graphical models and message-passing algorithms. We describe how message-passing algorithms relate to dynamic programming. We discuss a number of implementational issues.

Chapter 3. In this chapter, we discuss message-passing algorithms in the context of resource allocation problems. This is a broad and important class of problems in engineering and operations research that includes the network rate allocation example of Section 1.2.1. We propose message-passing as a framework for decentralized management. We demonstrate that that messages can be interpreted as a generalization of shadow prices, and are equivalent to shadow prices for convex problems. We argue that messages, however, provide a richer approximation to decision-making externalities than prices, and thus can provide superior solutions for nonconvex problems.

Chapter 4. In this chapter, we consider the asynchronous convergence of message-passing algorithms for unconstrained convex optimization problems. The results

we obtain are the strongest known convergence results for message-passing algorithms in this setting and generalize a number of prior results. We demonstrate that a key sufficient condition for this convergence is the notion of scaled diagonal dominance. This has previously been recognized as important in the asynchronous convergence of coordinate descent and gradient descent.

Chapter 5. In this chapter, we consider the application of message-passing algorithms to the distributed consensus problem of Section 1.2.3. We provide a theoretical analysis of the rate of convergence of message-passing. We argue that the convergence time of message-passing scales better than a competitive class of iterative, local search methods known as linear consensus algorithms.

Chapter 6. In this chapter, we offer some concluding remarks, and discuss directions for future work.

2

MESSAGE-PASSING ALGORITHMS

In this chapter, we introduce graphical models and message-passing algorithms in an optimization setting. Algorithms of this type are known as min-sum algorithms, and we will use these terms interchangeably.

For ease of exposition, we will focus principally on the special case of pairwise graphical models, where each factor in the decomposition of the objective involves at most two decision variables. Pairwise graphical models are introduced in Section 2.1. In Section 2.2, we consider the special case of a series graph, where the min-sum algorithm is equivalent to dynamic programming. In Section 2.3, we introduce the min-sum algorithm for arbitrary pairwise graphical models, and discuss a number of implementational issues. In Section 2.4, we present the min-sum algorithm for higher-order graphical models. Finally, in Section 2.5, we discuss nonserial dynamic programming, which is an alternative approach for solving optimization problems with graphical structure.

2.1 Pairwise Graphical Models

A pairwise graphical model is an optimization problem characterized by an undirected graph (V, E). There is a decision variable x_i associated with each vertex $i \in V$. Each decision variable takes values in a set $\mathcal{X}$. Denote by $x \in \mathcal{X}^V$ the vector of all decision variables,

$$x \triangleq (x_i : \ i \in V).$$

For each vertex $i \in V$, there is an extended real-valued function $f_i : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$. This function is referred to as a *single-variable factor*. Similarly, for each edge $(i,j) \in E$, there is an extended real-valued function $f_{ij} : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$. Each such function is referred to as a *pairwise factor*. The optimization problem is defined to be

$$\begin{array}{lll} \underset{x}{\text{minimize}} & F(x) \triangleq & \sum_{i \in V} f_i(x_i) + \sum_{(i,j) \in E} f_{ij}(x_i, x_j) \\ \text{subject to} & & x_i \in \mathcal{X}, \qquad \forall\, i \in V. \end{array} \tag{2.1}$$

Such an optimization problem is called a *pairwise graphical model*, since each term in the decomposition of the objective function involves at most a pair of decision variables.

In general, we will always assume that the graph (V, E) is connected. This is without loss of generality as, otherwise, the optimization problem will decompose into a collection of smaller problems, one for each connected component of the graph. These subproblems are completely independent and can be handled individually. Further, note that it is a trivial generalization to allow each decision variable x_i to take values in different domain $\mathcal{X}_i$. For simplicity, however, we will assume there is a common domain $\mathcal{X}$.

For the discussion that follows, it is convenient to define notation for the neighborhood structure of the graph (V, E). For each vertex $i \in V$, we define $N(i) \subset V$ to be the collection of neighboring vertices,

$$N(i) \triangleq \{j \in V : \ (i,j) \in E\}.$$

Denote the set of edges with direction distinguished by

$$\vec{E} \triangleq \{(i,j) \in V \times V : \ i \in N(j)\}.$$

As an example, consider the optimization program

$$\underset{x \in \mathcal{X}^4}{\text{minimize}} \ \ f_{12}(x_1, x_2) + f_{13}(x_1, x_3) + f_{14}(x_1, x_4) + f_{24}(x_2, x_4) + f_{34}(x_3, x_4). \tag{2.2}$$

The corresponding graph (V, E) is shown in Figure 2.1. Note that, in the absence of single-variable factors in the decomposition of (2.2), it is implicitly assumed that, for each $i \in V$, $f_i(\cdot) \triangleq 0$.

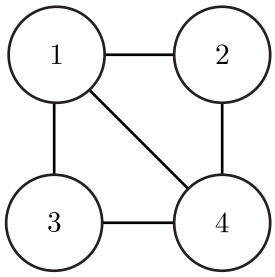


Figure 2.1 The graph corresponding to the example (2.2).

2.2 Dynamic Programming

Dynamic programs are a class of common optimization problems that fall within the framework of pairwise graphical models. Consider the following example of a finite horizon, deterministic dynamic program:

$$\underset{x \in \mathcal{X}^n}{\text{minimize}} \sum_{i=1}^{n} f_i(x_i) + \sum_{i=1}^{n-1} f_{i,i+1}(x_i, x_{i+1}). \tag{2.3}$$

Here, there is a decision variable for each of n time stages, and the objective decomposes into functions of a single decision, and functions of each pair of consecutive decisions. The corresponding graph is that of a chain (or series) of n vertices, as illustrated in Figure 2.2.

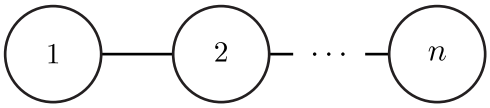


Figure 2.2 The graph corresponding to a finite horizon, deterministic dynamic program.

Rather than brute force minimization over an n-dimensional space, dynamic programming [13] offers a way to solve the optimization problem (2.3) by a series

of one-dimensional minimizations. The ith coordinate of an optimal solution can be determined by the single-variable optimization problem

$$\underset{x_i \in \mathcal{X}}{\text{minimize}} \;\; f_i(x_i) + \mathbb{I}_{\{i>1\}} J^*_{i-1\to i}(x_i) + \mathbb{I}_{\{i<n\}} J^*_{i+1\to i}(x_i),$$

where,

$$J^*_{i-1\to i}(x_i) \triangleq \min_{x_1,\ldots,x_{i-1}} \; \sum_{j=1}^{i-1} f_j(x_j) + \sum_{j=1}^{i-1} f_{j,j+1}(x_j, x_{j+1}), \quad \forall\, 1 < i \leq n, \tag{2.4}$$

$$J^*_{i+1\to i}(x_i) \triangleq \min_{x_{i+1},\ldots,x_n} \; \sum_{j=i+1}^{n} f_j(x_j) + \sum_{j=i}^{n-1} f_{j,j+1}(x_j, x_{j+1}), \quad \forall\, 1 \leq i < n. \tag{2.5}$$

Here, the "cost-to-go" functions $J^*_{i-1\to i}(\cdot)$ and $J^*_{i+1\to i}(\cdot)$, illustrated in Figure 2.3, capture the impact of the decision at vertex i to the objective function along the chain to the left and to the right of vertex i, respectively.

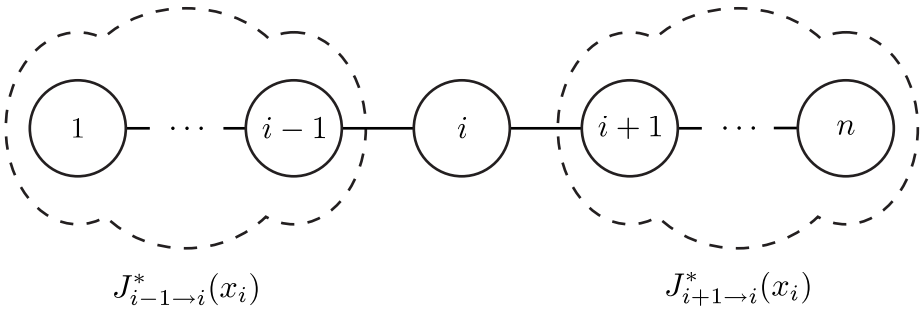


Figure 2.3 Cost-to-go functions in a dynamic programming solution.

The functions $J^*_{i-1\to i}(\cdot)$ and $J^*_{i+1\to i}(\cdot)$ can be can be decomposed recursively according to

$$J^*_{i-1\to i}(x_i) = \min_{x_{i-1}} \; f_{i-1}(x_{i-1}) + f_{i-1,i}(x_{i-1}, x_i) + \mathbb{I}_{\{i>2\}} J^*_{i-2\to i-1}(x_{i-1}), \tag{2.6}$$

$$J^*_{i+1\to i}(x_i) = \min_{x_{i+1}} \; f_{i+1}(x_{i+1}) + f_{i,i+1}(x_i, x_{i+1}) + \mathbb{I}_{\{i<n-1\}} J^*_{i+2\to i+1}(x_{i+1}). \tag{2.7}$$

This decomposition suggests a simple, iterative procedure for the computation of these functions: begin at time zero with a collection of functions

$$\{J^{(0)}_{i-1\to i}(\cdot),\; J^{(0)}_{i+1\to i}(\cdot)\},$$

that is initialized arbitrarily. At each time $t \geq 0$, these functions are updated according according to

$$J_{i-1\to i}^{(t+1)}(x_i) \triangleq \min_{x_{i-1}} \; f_{i-1}(x_{i-1}) + f_{i-1,i}(x_{i-1}, x_i) + \mathbb{I}_{\{i>2\}} J_{i-2\to i-1}^{(t)}(x_{i-1}), \tag{2.8}$$

$$J_{i+1\to i}^{(t+1)}(x_i) \triangleq \min_{x_{i+1}} \; f_{i+1}(x_{i+1}) + f_{i,i+1}(x_i, x_{i+1}) + \mathbb{I}_{\{i<n-1\}} J_{i+2\to i+1}^{(t)}(x_{i+1}). \tag{2.9}$$

These updates can be interpreted as a parallel and iterative implementation of standard (deterministic) dynamic programming updates [13]. For each vertex i at each time t, define the single-variable objective

$$b_i^{(t)}(x_i) \triangleq f_i(x_i) + \mathbb{I}_{\{i>1\}} J_{i-1\to i}^{(t)}(x_i) + \mathbb{I}_{\{i<n\}} J_{i+1\to i}^{(t)}(x_i).$$

Each vertex i can then generate an estimate of the ith coordinate of an optimal solution according to

$$x_i^{(t)} \in \mathcal{X}_i^{(t)} \triangleq \operatorname*{argmin}_{x_i} \; b_i^{(t)}(x_i). \tag{2.10}$$

It is straightforward to see that, after $t \geq n$ iterations, the cost-to-go function iterates will have converged to the true cost-to-go functions. Denote the set of optimal solutions to the original program (2.3) by $\mathcal{X}^*$. Then, the projection of the elements of $\mathcal{X}^*$ onto the ith coordinate are in one-to-one correspondence with the elements of the set $\mathcal{X}_i^{(t)}$ of optimal solutions to (2.10), that is

$$\{x_i : \; \exists \; x^* \in \mathcal{X}^* \text{ with } x_i = x_i^*\} = \mathcal{X}_i^{(t)}.$$

Our goal is to construct an optimal solution $x^* \in \mathcal{X}^*$. If this solution is unique, then for each ith coordinate, the solution $x_i^{(t)}$ of single-variable optimization problem (2.10) will be unique. Thus, x^* can be determined component-wise in a parallel fashion through the estimates $\{x_i^{(t)}\}$.

On the other hand, if there are multiple optimal solutions, there will exist a vertex i for which the solution to the single-variable problem (2.10) is not unique. Such ties must be broken in a consistent fashion across the vertices in order to construct an optimal solution x^*. In such cases, a single optimal solution x^* be

sampled from the set of all possible solutions $\mathcal{X}^*$ by a recursive procedure. Here, some distinguished vertex is chosen. Without loss of generality, set this to be vertex 1. An arbitrary choice of x_1^* is made from the set $\mathcal{X}_1^{(t)}$. Then, sequentially, for vertices $i = 2, \ldots, n$, a choice for x_i^* is made by solving the original optimization problem (2.3) assuming that $x_j = x_j^*$, for $j < i$. These choices can be decomposed recursively so that x_i^* is chosen arbitrarily from the set

$$\operatorname*{argmin}_{x_i} f_i(x_i) + f_{i-1,i}(x_{i-1}^*, x_i) + \mathbb{I}_{\{i<n\}} J_{i+1\to i}^{(t)}(x_i). \tag{2.11}$$

Note that the presentation here has focused on a parallel variation of dynamic programming, where all the cost-to-go functions are simultaneously computed in an iterative fashion. In a more traditional application of dynamic programming, the cost-to-go functions $\{J_{i+1\to i}^*(\cdot)\}$ would be computed sequentially by (2.7), as $i = n-1, \ldots, 1$, in a "backward pass". The coordinate solutions $\{x_i^*\}$ can then be computed sequentially by (2.11), as $i = 1, \ldots, n$, in a "forward pass".

2.3 The Min-Sum Algorithm

The idea of the min-sum algorithm is simply to extend the iterative dynamic programming updates described in Section 2.2 from chains to arbitrary graphs.

Consider a pairwise graphical model associated with a graph (V, E). For vertex i with a neighboring vertex $j \in N(i)$, we define a sequence of "messages" $\{J_{i\to j}^{(t)}(\cdot)\}$ from vertex i to vertex j. These messages are initialized arbitrarily. They evolve over time in an iterative fashion, according to an update rule that incorporates information received by vertex i from its set of neighboring vertices $N(i)$ *excluding* vertex j, as depicted in Figure 2.4. The update rule is

$$J_{i\to j}^{(t+1)}(x_j) \triangleq \min_{x_i} \ \kappa_{i\to j}^{(t)} + f_i(x_i) + f_{ij}(x_i, x_j) + \sum_{u\in N(i)\setminus j} J_{u\to i}^{(t)}(x_i). \tag{2.12}$$

Here, $\kappa_{i\to j}^{(t)}$ is a normalization term which is allowed to vary from message to message.

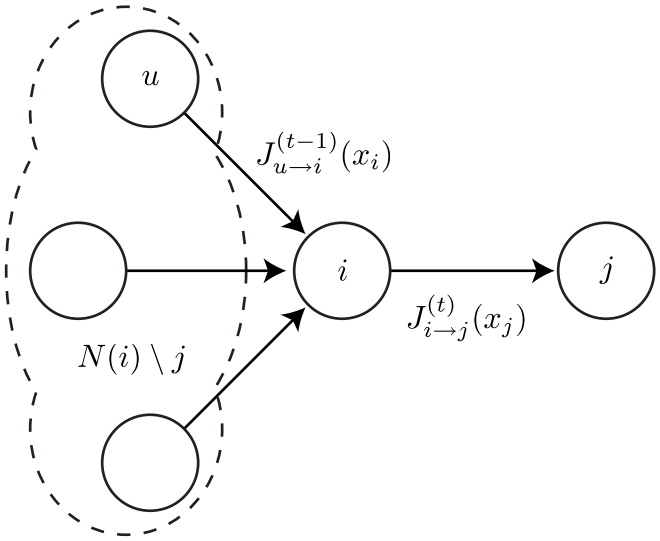


Figure 2.4 The computation of a min-sum update.

Given the collection of messages received at a vertex j, a local objective function is defined according to

$$b_j^{(t)}(x_j) \triangleq f_j(x_j) + \sum_{u \in N(j)} J_{u\to j}^{(t)}(x_j). \tag{2.13}$$

An estimate $x_j^{(t)}$ of the jth coordinate of the optimal solution can then be computed by solving the optimization program

$$x_j^{(t)} \in \mathcal{X}_j^{(t)} \triangleq \operatorname*{argmin}_{x_j} \; b_j^{(t)}(x_j). \tag{2.14}$$

2.3.1 Relationship to Dynamic Programming

Assume, for the moment, that the normalization terms $\kappa_{i\to j}^{(t)}$ are set to zero. Then, in the case of the deterministic dynamic program (2.3), the min-sum update equation (2.12) and the dynamic programming update (2.8)–(2.9) coincide. Hence, the message iterates $\{J_{i\to j}^{(t)}(\cdot)\}$ will converge to the cost-to-go function $\{J_{i\to j}^{*}(\cdot)\}$,

and each jth coordinate estimate $x_j^{(t)}$ will yield the jth coordinate of an optimal solution.

The argument above extends to the case where the graph (V, E) is singly-connected (that is, a connected graph with no cycles). Here, it is straightforward to see that minimizing the function $b_j^{(t)}(\cdot)$ is equivalent to estimating the optimal value of x_j as reflected only in the terms of the objective function which are distance t or closer to vertex j. Thus, after a number of iterations equal to the diameter of the tree, the functions $\{b_j^{(t)}(\cdot)\}$ determine component values of optimal solutions to the original problem.

This algorithm, however, can also be applied to graphs with cycles. Such "loopy" variations, which are of primary interest to us, are not justified by dynamic programming arguments.

2.3.2 Normalization

Note that the messages influence the estimates produced by the algorithm only through the minimization (2.14). Hence, only *relative* values of the messages matter, and the choice of normalization constants does not affect the output of the algorithm.

However, in graphs with cycles, in the absence of normalization, there may be no collection of messages which is a fixed point to the min-sum update equation (2.12). Thus, normalization does influence convergence of messages and the numerical stability of the algorithm. We will make a choice of the normalization constant $\kappa_{i\to j}^{(t)}$ in (2.12) so that, given a distinguished state $0 \in \mathcal{X}$, $J_{i\to j}^{(t)}(0) = 0$.

2.3.3 Distributed and Asynchronous Implementation

One important characteristic of the min-sum algorithm is that it naturally lends itself to a distributed and asynchronous implementation. In such a setting, we imagine that there is a collection of processors, each associated with a vertex $i \in V$, and responsible for the determining the optimal value of the decision variable

x_i. Each processor is allowed to communicate only to neighboring processors, as determined by the edges of the graph.

Assume that each processor i keeps track of incoming messages from its neighboring processors. It uses these messages to occasionally compute new outgoing messages to each neighboring processor $j \in N(i)$. Define $T^{i\to j}$ to be the set of times at which new messages are computed. The processor i will occasionally communicate with each neighboring processor $j \in N(i)$, and transmit the appropriate outgoing message. Define $0 \leq \tau_{i\to j}(t) \leq t$ to be the time at which the most recent message received by processor j, at or before time t, from processor i was computed. Thus, the quantity

$$t - \tau_{i\to j}(t) \geq 0$$

represents the delay experienced at time t in the communications from processor i to processor j.

The messages evolve according to

$$J_{i\to j}^{(t+1)}(x_j) \triangleq \min_{x_i} \ \kappa_{i\to j}^{(t)} + f_i(x_i) + f_{ij}(x_i, x_j) + \sum_{u\in N(i)\setminus j} J_{u\to i}^{(\tau_{u\to i}(t))}(x_i), \tag{2.15}$$

if $t \in T^{i\to j}$. Thus, when processor i updates its message to processor j, it uses the most recent messages it has received from other processors. At times $t \notin T^{i\to j}$, the messages are not updated,

$$J_{i\to j}^{(t+1)}(x_j) \triangleq J_{i\to j}^{(t)}(x_j). \tag{2.16}$$

Each processor $j \in V$ will maintain a sequence of estimates $\{x_j^{(t)}\}$ of the optimal value of the decision variable x_j. These estimates will be recomputed at a set of times T^j. For each time $t \in T^j$, the processor j can define a local objective function by using the most recent message received from each neighbor, according to

$$b_j^{(t)}(x_j) \triangleq f_j(x_j) + \sum_{u\in N(j)} J_{u\to j}^{(\tau_{u\to i}(t))}(x_j). \tag{2.17}$$

The estimate $x_j^{(t)}$ is computed by solving the optimization program

$$x_j^{(t)} \in \operatorname*{argmin}_{x_j} \; b_j^{(t)}(x_j). \tag{2.18}$$

For all other times $t \notin T^j$, the estimate remains unchanged,

$$x_j^{(t)} \triangleq x_j^{(t-1)}. \tag{2.19}$$

When analyzing the asynchronous algorithm (2.15)–(2.19), we will make the assumption of *total asynchronism* [14]:

Assumption 2.1 (Total Asynchronism) *Assume that:*

(i) For each $i \in V$ and $j \in N(i)$, the set $T^{i\to j}$ is infinite.

(ii) For each $j \in V$, the set T^j is infinite.

(iii) For each $u \in V$, $i \in N(u)$, and $j \in N(i) \setminus u$, if $\{t_k\}$ is a sequence in $T^{i\to j}$ tending to infinity, then

$$\lim_{k\to\infty} \tau_{u\to i}(t_k) = \infty.$$

(iv) For each $j \in V$ and $i \in N(j)$, if $\{t_k\}$ is a sequence in T^j tending to infinity,

$$\lim_{k\to\infty} \tau_{i\to j}(t_k) = \infty.$$

Total asynchronism is a very mild assumption, and is perhaps the least restrictive useful model of asynchronous computation. It guarantees that each message and estimate is updated infinitely often, that each processor eventually communicates with each neighboring processor, and that old information is eventually purged from the system. It allows for arbitrary delays in communication, and even the out-of-order arrival of messages between processors.

In this setting, it is clear that fixed points of the asynchronous algorithm (2.15)–(2.19) coincide with fixed points of the original, synchronous algorithm (2.12)–(2.14).

2.3.4 Nonuniqueness of Estimates

If each estimate in $x_j^{(t)}$ in (2.14) is unique, then it is natural to construct an estimate of an overall solution to the original optimization program (2.1) in a component-wise fashion in parallel. If these estimates are not uniquely defined, however, a procedure is needed to consistently break ties.

One possibility, in the spirit of the method described for dynamic programs in Section 2.2, is to pick a distinguished vertex j, make an arbitrary choice of $x_j^{(t)}$ from $\mathcal{X}_j^{(t)}$, and to consider the original program (2.1) when the jth coordinate is fixed to $x_j^{(t)}$. This way, the original program is reduced to a smaller program with one less variable. When the graph (V, E) is singly-connected, this can be done according to a recursive decomposition as described in Section 2.2. For graphs with cycles, however, the recursive decomposition does not apply.

Instead, typically the min-sum algorithm is run for a larger number of iterations t, or until the messages have converged. Then, the distinguished vertex j is determined by finding the decision variable that is most "biased" according to the messages at time t. One way to do this, for example, might be to computed

$$\operatorname*{argmax}_j \left(\max_{x_j} \ b_j^{(t)}(x_j) - \min_{x_j} \ b_j^{(t)}(x_j) \right).$$

Once a value for x_j is chosen from $\mathcal{X}_j^{(t)}$, message-passing can be applied to the reduced problem.

Alternatively, in some cases such as the example described in Section 3.6.3, problem specific tie-breaking rules can be developed. In general, however, the question of how to break ties that occur in message-passing is open.

2.4 Higher-Order Graphical Models

Higher-order models, where factors may involve more than two decision variables, are naturally characterized by hypergraphs. A hypergraph $(V, \mathcal{C})$ consists of a set of vertices V and a set of hyperedges $\mathcal{C}$. Each hyperedge $C \in \mathcal{C}$ is a nonempty

subset of the vertices, that is, $C \subset \mathcal{V}$. A decision variable $x_i \in \mathcal{X}$ is associated with each vertex $i \in V$. A factor $f_C : \mathcal{X}^C \to \mathbb{R} \cup \{+\infty\}$ is associated with each hyperedge $C \in \mathcal{C}$. The optimization problem takes the form

$$\begin{array}{lll} \underset{x}{\text{minimize}} & F(x) \triangleq & \sum_{C \in \mathcal{C}} f_C(x_C) \\ \text{subject to} & & x_i \in \mathcal{X}, \qquad \forall\ i \in V. \end{array} \tag{2.20}$$

The hypergraph $(V, \mathcal{C})$ can be visualized as a bipartite graph known as a *factor graph*. Such a graph is shown in in Figure 2.5. There are vertices represented by circles corresponding to each decision variable. There are vertices represented by squares corresponding to each factor. There is an edge between the vertices associated with $i \in V$ and $C \in \mathcal{C}$ iff $i \in C$. For each variable $i \in V$, denote by ∂i the set of factors that involve that variable, i.e., $\partial i \triangleq \{C \in \mathcal{C} : i \in C\}$.

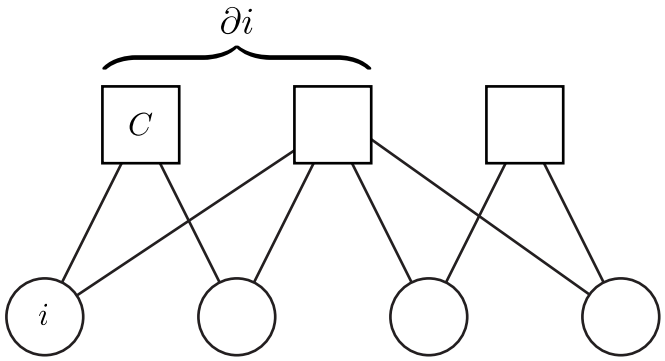


Figure 2.5 A factor graph. Circular vertices represent decision variables, while square vertices represent factors of the objective function. There is an edge between each decision variable and the factors that it participates in.

2.4.1 The Min-Sum Algorithm

The min-sum algorithm for pairwise graphical models, described in Section 2.3, was derived through dynamic programming arguments, under the assumption that the graph was a tree. The same derivation can be applied in the case of higher-order graphical models.

As in the pairwise case, the min-sum algorithm operates in an iterative fashion via the exchange of messages. In this case, for each variable $i \in V$ and each associated factor $C \in \partial i$, there is a message $J_{i \to C} : \mathcal{X} \to \mathbb{R}$ from i to C. Similarly, for each factor $C \in \mathcal{C}$ and each associated variable $i \in C$, there is a message $J_{C \to i} : \mathcal{X} \to \mathbb{R}$ from C to i. Given a collection of all messages $\{J^{(t)}_{i \to C}(\cdot),\ J^{(t)}_{C \to i}(\cdot)\}$ at time t, new messages are computed according to

$$J^{(t+1)}_{i \to C}(x_i) \triangleq \sum_{C' \in \partial i \setminus C} J^{(t)}_{C' \to i}(x_i) + \kappa^{(t)}_{i \to C}, \tag{2.21}$$

$$J^{(t+1)}_{C \to i}(x_i) \triangleq \min_{x_{C \setminus i}} \ f_C(x_C) + \sum_{i' \in C \setminus i} J^{(t+1)}_{i' \to C}(x_{i'}) + \kappa^{(t)}_{C \to i}. \tag{2.22}$$

Local objective functions and estimates of the optimal solution are defined by

$$b^{(t)}_i(x_i) \triangleq f_i(x_i) + \sum_{C \in \partial i} J^{(t)}_{C \to i}(x_i),$$
$$x^{(t)}_i \in \operatorname*{argmin}_{y_i} b^{(t)}_i(y_i).$$

2.5 Nonserial Dynamic Programming

It is worth pointing out that graphical models of the form (2.20) have been known historically in the optimization community as *nonserial dynamic programs* [12]. Over the past few decades, however, this formulation has received virtually no attention from the mathematical programming community. Perhaps this is due to a perception that the formulation is computationally unmanageable.

One conventional approach to exploiting graphical structure is that of tree decomposition [75, 76]. In this method, the nonserial dynamic program is converted to an equivalent program (sometimes known as a "junction tree") whose underlying graph is a tree. Techniques such as dynamic programming or variable elimination can then be applied to this tree in order to obtain exact solutions to the original optimization problem. Such methods have been proposed for a number of difficult combinatorial optimization problems (see, for example, [4, 17, 16, 43, 42]).

The complexity of such a procedure, however, is exponentially dependent on a parameter of the original graph known as the treewidth. Many practical nonserial dynamic programs have a large treewidth, and hence tree decomposition methods often are not computationally tractable.

3

RESOURCE ALLOCATION

Consider a system consisting of a set of activities and a set of resources. Each activity contributes utility to an overall system objective, as a function of the resources allocated to it, and each resource is of limited supply. The system manager's decision problem is to allocate resources between the activities, so as to maximize overall utility. This resulting optimization program, called a *resource allocation problem*, has an objective function and constraints that are additively separable. It is one of the oldest and most well-studied problems in operations research, economics, and engineering.

In this chapter, we study the application of ideas from message-passing to resource allocation problems. In such problems, decentralized methods are important for a number of reasons which we will shortly discuss. Hence, message-passing algorithms are particularly relevant. Moreover, resource allocation problems have been studied extensively using tools from the classical theory of convex analysis. By considering message-passing algorithms in this context, we seek to understand how they relate to conventional methods and, thus, the benefits they offer.

The balance of the chapter is organized as follows: In Section 3.1, we provide a high-level introduction to decentralized decision making in resource allocation problems, and outline the contributions of this chapter. In Section 3.2, we describe the resource allocation problem formulation. In Section 3.3, we describe the decision externalities that occur because of decentralization. In Section 3.4, we define the concept of a message-passing equilibrium, and compare the optimality properties of the message-based incentives with those of incentives based on

shadow prices or Lagrange multipliers. In Section 3.5, we describe a distributed asynchronous algorithm for computing message-passing equilibria. Finally, in Section 3.6, we discuss the application of message-passing to a network rate allocation problem. Proofs are provided in Section 3.7.

3.1 Decentralized Decision Making

We are interested in *decentralized* decision making methods for resource allocation. Such methods decompose the problem across the collection of agents that participate in the system. The spirit here is to allow activity managers, each responsible for a particular activity, to make their own resource consumption decisions. These decisions cannot be made in isolation, however. Since resources may be profitably used by other activities, consumption decisions by a single activity manager have an impact across the entire system. Decentralized methods address these decision externalities via coordination signals, or incentives[1], that influence resource consumption decisions. These incentives serve to align the objective of each individual activity manager to that of the system.

3.1.1 Benefits of Decentralized Methods

One benefit of decentralized methods is that they allow for greater flexibility in the management of complex systems. This is illustrated in the following example:

Example 3.1 (Organizational Management) *Consider a large and complex firm. Activities represent divisions of the firm, and resources represent inputs to the processes of the firm, such as capital or raw materials, that are of limited supply. The firm's resource allocation problem is to optimize the distribution of the resources across the divisions. Each division may, in turn, be faced with its own complicated*

[1]Note that, in this thesis, we are not considering "incentives" in a game theoretic sense, but rather as a coordination mechanism. We are assuming that activity managers are myopic with respect to the incentives they are provided, and do not seek to manipulate these incentives through strategic behavior. This is as in a price-taking or competitive equilibrium setting.

decision making process. Given an allocation of resources, the benefit generated by a division's activity may entail optimization of a large number of decisions that govern how the activity is conducted. Any model of the division that is tractable from the perspective of a central planner will necessarily be simplified or abstract. As such, the resource allocation decisions made by a central planner can constrain activities in ways that prevent the beneficial reallocation of resources between activities.

An alternative to the centralized micromanagement of resources is to have resource consumption decisions made by each individual division. The activity managers will have the greatest expertise in and knowledge of their particular activities. Further, over time, the activities may be changing, or the managers may be learning how to better conduct their activities. Hence, activity managers are in the best position to accurately model and understand their resource needs on an ongoing basis. By having individual divisions make their own resource consumption decisions, decentralized methods allow for greater management flexibility, and more robust and efficient decision making.

Decentralized methods provide further benefits by economizing communication costs and distributing information processing tasks. This allows for their use in many settings, such as the following, where centralized solutions have prohibitive communication and computational requirements:

Example 3.2 (Network Rate Control) *Consider a communications network consisting of a set of links (resources), and a set of users (activities). Each user wishes to transmit data across a particular path (subset of links) in the network, and generates utility as a function of the transmission rate allocated to it. Each link in the network is capable of transmitting data at some finite capacity. The network manager's problem is to allocate the capacity along each link among the users requiring service from the link, so as to maximize the overall utility.*

In such a network, the users and links are geographically distributed and physically disparate. A central planner would require a global view of the network.

This would entail significant additional communication that may degrade the performance of the network. Further, a central planner would require computational resources commensurate with the size of the network. Decentralized methods, on the other hand, allow users and links to coordinate their respective consumption and allocation decisions by purely local communication that occurs alongside the regular flow of network traffic. Neither the agents nor the network manager require knowledge of the entire network. Further, since the computational burden is shifted to the agents that comprise the network, the network manager does not require additional computational resources.

3.1.2 Priced-Based Methods

In the case where the utility functions are concave, called the *convex resource allocation problem*, the classical theory of convex optimization establishes shadow prices (Lagrange multipliers) as proxies for decentralization. Given a proper set of prices for resources, each activity manager can optimize resource consumption so as to maximize the utility generated by the activity minus the cost (as reflected through prices) of the consumed resources, so that the resulting decision will be optimal for the system manager's problem. Price-based methods for decentralized resource allocation have been developed as far back as the 1950's, dating to the pioneering work of Arrow, Hurwicz, and others [5]. Such methods have the following benefits:

1. *A tractable representation of externalities that leads to system-optimal behavior.*

   Prices provide a linear representation of externalities, and concisely summarize the impact of decisions across the system. They enable each activity manager to align their objective with that of the system manager.

2. *Distributed asynchronous algorithms for computing prices and allocations.*

   Optimal prices and allocations can be computed iteratively via gradient methods. These local search methods require only communication between

activity managers, which make resource consumption decisions, and resource managers, which determine prices. Further, each activity manager needs only to communicate with the resource managers for resources it requires. Neither communication with nor even knowledge of other activities and resources is necessary, nor is any other global coordination or synchronization required.

In convex resource allocation problems, fixed prices can provide appropriate incentives to induce system-optimal decisions within activities. This is not generally true for nonconvex problems, where there may be no set of prices which supports a globally optimal allocation. Nonconvexities appear in many practical problem instances for a host of reasons. The underlying resources may be discrete and indivisible. The activities may have increasing returns to scale, or inelastic demand for resources. In such cases, price-based decentralized algorithms may converge to local optima, or may fail to converge at all.

3.1.3 Contributions of This Chapter

In this chapter, we consider prices that vary across activities and consumption levels. We refer to such nonlinear price functions as *messages*, as they can be viewed as incentives communicated between resource managers and activity managers. Message-based incentives allow for a richer description of externalities than prices, while still maintaining computational tractability. We argue that messages extend many of the benefits of prices to nonconvex resource allocation problems. The contributions of this chapter are as follows:

1. *We propose a new equilibrium concept for message-based incentives.*
   We define a set of equilibrium message-based incentives as the fixed points of a message-passing operator. We demonstrate that, under broad technical conditions, these equilibria exist.
2. *We demonstrate that messages lead to system-optimal behavior for convex problems.*

We demonstrate that in the convex case, message-passing equilibria lead to system-optimal behavior. Indeed, in this case, messages are locally equivalent to prices: the marginal incentives provided by a set of equilibrium messages at the optimal allocation are precisely optimal shadow prices.

3. *We argue that messages yield allocations superior to prices for nonconvex problems.*

   For nonconvex problems, in general, message-based incentives will not guarantee system-optimal allocations. This is not surprising, because this class of problems includes many which are provably intractable. Any method which guarantees global optimality is not likely to be of practical use in large scale problems. Allocations resulting from message-based incentives will, however, satisfy a property which precludes the improvement of the system objective under certain types of transfers of resources between activities. This property is stronger than the analogous local optimality guarantees which can be made for price-based incentives. Further, we numerically demonstrate the superiority of message-based incentives by considering a well-studied inelastic network rate control example.

4. *We describe a distributed asynchronous algorithm for computing messages and allocations.*

   Equilibrium messages can be computed via a successive approximations procedure. We show how this procedure decomposes into purely local communication between activity and resource managers. In the inelastic rate control example, this takes a particularly simple form where the algorithm operates alongside the normal flow of network traffic, and appends a single real number to each data packet.

3.2 Problem Formulation

Consider the following prototypical resource allocation problem: a set of resources $\mathcal{R}$, each of finite capacity, is to be allocated among a set of activities $\mathcal{A}$. Each

activity $a \in \mathcal{A}$ depends on some subset $\partial a \subseteq \mathcal{R}$ of the resources. For each a and each $r \in \partial a$, denote by $x_{ar} \geq 0$ the decision variable representing the quantity of resource r to be allocated to activity a. Denote the allocation decisions by $x \triangleq (x_{ar} : \ a \in \mathcal{A}, \ r \in \partial a)$. Denote by $x_{\partial a} \triangleq (x_{ar} : \ r \in \partial a$ the consumption bundle for activity a. A utility function $u_a(\cdot)$ specifies the contribution $u_a(x_{\partial a}) \in \mathbb{R}$ of activity a to the overall system objective, as a function of the allocation $x_{\partial a}$ it receives.

For each resource r, denote by $\partial r \triangleq \{a \in \mathcal{A} : \ r \in \partial a\} \subseteq \mathcal{A}$ the set of activities which depend on resource r. Denote by $x_{\partial r} \triangleq (x_{ar} : \ a \in \partial r)$ the vector of allocations of resource r. There is a finite quantity $b_r > 0$ of each resource r available, hence we require that $x_{ar} \in \mathcal{X}_r \triangleq [0, b_r]$, for all $a \in \partial r$, and that

$$\sum_{a \in \partial r} x_{ar} \leq b_r.$$

The relationships between activities and resources can be conveniently encoded using a graphical representation:

Definition 3.1 (Dependency Graph) *Define the dependency graph $\mathcal{D}$ to be an undirected bipartite graph consisting of vertices corresponding to the activities $\mathcal{A}$ and the resources $\mathcal{R}$. An edge (a, r) is present if and only if activity a depends on resource r, that is, if $a \in \partial r$.*

Note that dependency graphs are closely related to the hypergraphs that were employed to describe graphical structure in Chapter 2. Dependency graphs are specialized, however, to structure of resource allocation problems. In these problems, we imagine that the system is comprised of specific agents that are responsible for each activity and resource. These agents are modeled as the atomic decision-makers of the system, thus they most naturally correspond to vertices.

An optimal allocation is determined by solving the following program:

$$\begin{array}{lll} \text{maximize} \quad U(x) \triangleq & \sum_{a \in \mathcal{A}} u_a(x_{\partial a}) & \\ \text{subject to} & \sum_{a \in \partial r} x_{ar} \leq b_r, & \forall \ r \in \mathcal{R}, \\ & x_{ar} \in \mathcal{X}_r, & \forall \ a \in \mathcal{A}, \ r \in \partial a. \end{array} \tag{3.1}$$

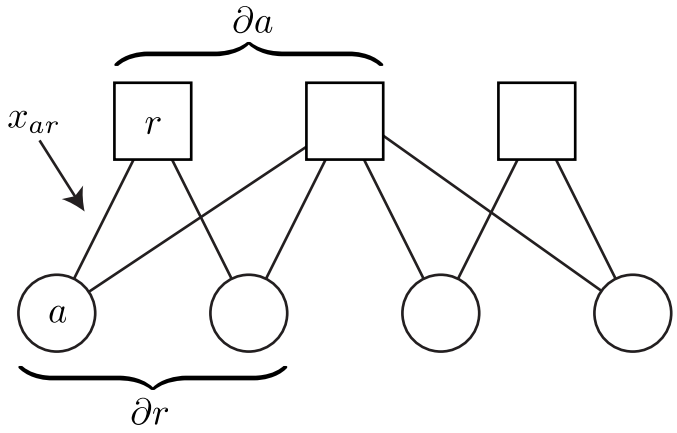


Figure 3.1 A dependency graph. Vertices in the graph correspond to activities and resources, edges in the graph correspond to decision variables.

The function $U(\cdot)$ is called the *system objective function*, and the problem (3.1) is called the *system manager's problem*.

If the utility functions are concave, this optimization problem can be addressed by methods of convex optimization, as we discuss in Section 3.3.1. Our primary motivation, however, is to consider cases where utility functions are not concave, as in the following example, which we revisit in Section 3.6.

Example 3.3 (Inelastic Rate Control) *Consider a communications network consisting of a set of links (resources), and a set of users (activities). Each user a wishes to transmit data across a particular path (subset of links) ∂a in the network. For each user a and each link $r \in \partial a$, the decision variable x_{ra} represents the data transmission rate on the link r that is allocated to the user a. Each link in the network is capable of transmitting data at some finite capacity.*

The overall transmission rate for a user is constrained by the minimum transmission rate it is allocated along all the links in its path. Each user a desires some minimum overall transmission rate $w_a > 0$. If the user is able to transmit at that rate, the user derives utility $z_a > 0$. Otherwise, the user derives 0 utility. Hence,

the utility function for user a takes the form

$$u_a(x_{\partial a}) = \begin{cases} z_a & \text{if, for each } r \in \partial a,\ x_{ar} \geq w_a, \\ 0 & \text{otherwise,} \end{cases}$$

which is not concave.

3.3 Decentralization and Externalities

Under a decentralized decision making scheme, individual activity managers make their own resource consumption decisions. These individual decisions impact the entire system since, as a resource is consumed by one activity, the quantity of the resource available for other activities is reduced. A coordination mechanism is required to address these decision externalities.

One very general way that this can be accomplished is as follows: for each activity a, consider the optimization problem

$$\begin{array}{lll} \text{maximize} & u_a(x_{\partial a}) + E_a(x_{\partial a}) & \\ \text{subject to} & x_{ar} \in \mathcal{X}_a, & \forall\ r \in \partial a. \end{array} \tag{3.2}$$

Here, the function $E_a(\cdot)$ is defined by

$$\begin{array}{lll} E_a(x_{\partial a}) \triangleq \text{maximize} & \sum_{a' \in \mathcal{A} \setminus a} u_{a'}(x_{\partial a'}) & \\ \text{subject to} & \sum_{a' \in \partial r} x_{a'r} \leq b_r, & \forall\ r \in \mathcal{R}, \\ & x_{a'r} \in \mathcal{X}_r, & \forall\ a' \in \mathcal{A} \setminus a,\ r \in \partial a'. \end{array} \tag{3.3}$$

Given a consumption decision $x_{\partial a}$ for user a, the quantity $E_a(x_{\partial a})$ is the optimized value of utility across the rest of the system. Relative values of $E_a(\cdot)$ exactly capture the impact of consumption decisions for the activity a to the rest of the system. This function can be used as an incentive to the activity manager, aligning the objective (3.2) of the activity manager and the objective (3.1) of the system manager.

In general, however, such a mechanism is not practical. The function $E_a(\cdot)$ can be an arbitrary multidimensional nonlinear function. It is not clear how to tractably represent or compute such an object, much less in a decentralized manner.

3.3.1 Concave Utility Functions

It is well-known that if utility functions are strictly concave, then the optimal allocation is unique and supported by a set of prices. In particular, there exists an allocation x^* and a price vector $p^* \in \mathbb{R}_+^{\mathcal{R}}$, such that x^* is the unique optimal solution to the system manager's problem (3.1), and each $x^*_{\partial a}$ is the unique maximizer of the optimization problem

$$\begin{array}{lll} \text{maximize} & u_a(x_{\partial a}) - \sum_{r \in \partial a} p^*_r x_{ar} & \\ \text{subject to} & x_{ar} \in \mathcal{X}_r, & \forall\ r \in \partial a. \end{array} \tag{3.4}$$

This program opens the door to decentralized management based on an incentive system. Instead of overseeing each activity's consumption, the manager of a resource can set a unit price and leave consumption decisions in the hands of activity managers. If the manager for activity a maximizes the utility his activity generates minus the cost of resources consumed, objectives are aligned and he chooses to consume exactly $x^*_{\partial a}$.

One way to interpret a price-based incentive system is as a linear and separable approximation to the true externalities. If the utility functions are concave, the solution of (3.2) is determined by first-order conditions. Hence, we need only to characterize the first-order behavior of $E_a(\cdot)$ around the optimal allocation $x^*_{\partial a}$. This behavior is captured by the shadow price vector p^*, and the price-based incentives in the optimization program (3.4).

Unfortunately, the preceding story does not generally apply when utility functions are nonconcave. Even if there is a unique optimal solution, there may be no price vector that supports it. There are models in which, for any set of prices, choices made by activity managers would not be optimal. The solution concept presented in the next section generalizes price-based incentives in a way that addresses this.

3.4 Solution Concept

Our solution concept involves a general class of incentives, which we refer to as *messages*. These messages are exchanged between managers for each activity and each resource. For each activity a, the activity manager receives a message from the resource manager for each resource $r \in \partial a$. This message is a function $V_{r\to a} : \mathcal{X}_r \to \mathbb{R}$. The quantity $V_{r\to a}(x_{ar})$ can be thought of as a penalty imposed on activity a for consuming x_{ar} units from the finite supply of resource r that is available.

Similarly, for each resource r, the resource manager receives a message from each activity manager corresponding to an activity $a \in \partial r$. This message is a function $V_{a\to r} : \mathcal{X}_r \to \mathbb{R}$. The quantity $V_{a\to r}(x_{ar})$ can be thought of as a benefit generated to the resource manager by allocating x_{ar} units from its finite supply to activity a.

The spirit here is to allow decisions to be made in a decentralized manner: for each activity a, the activity manager makes a consumption decision that optimizes

$$\begin{array}{lll} \text{maximize} & u_a(x_{\partial a}) + \sum_{r\in\partial a} V_{r\to a}(x_{ar}) & \\ \text{subject to} & x_{ar} \in \mathcal{X}_r, & \forall\ r \in \partial a. \end{array} \tag{3.5}$$

Comparing with (3.2), the messages received by the manager of an activity a can be viewed as an additively separable approximation to the true externalities,

$$E_a(x_{\partial a}) \approx \sum_{r\in\partial a} V_{r\to a}(x_{ar}). \tag{3.6}$$

This approximation is motivated by the case where the dependency graph $\mathcal{D}$ is a tree, that is, a graph with no cycles. In this case, the impact on the rest of the system that occurs when the activity consumes a particular quantity of a resource does not depend on the quantities of other resources consumed by the activity. Hence, the approximation (3.6) is exact. This is illustrated in Figure 3.2. There, the optimization problem (3.3) for the externalities of activity a decomposes so that

$$E_a(x_{ar_1}, x_{ar_2}, x_{ar_3}) = V_{r_1\to a}(x_{ar_1}) + V_{r_2\to a}(x_{ar_2}) + V_{r_3\to a}(x_{ar_3}).$$

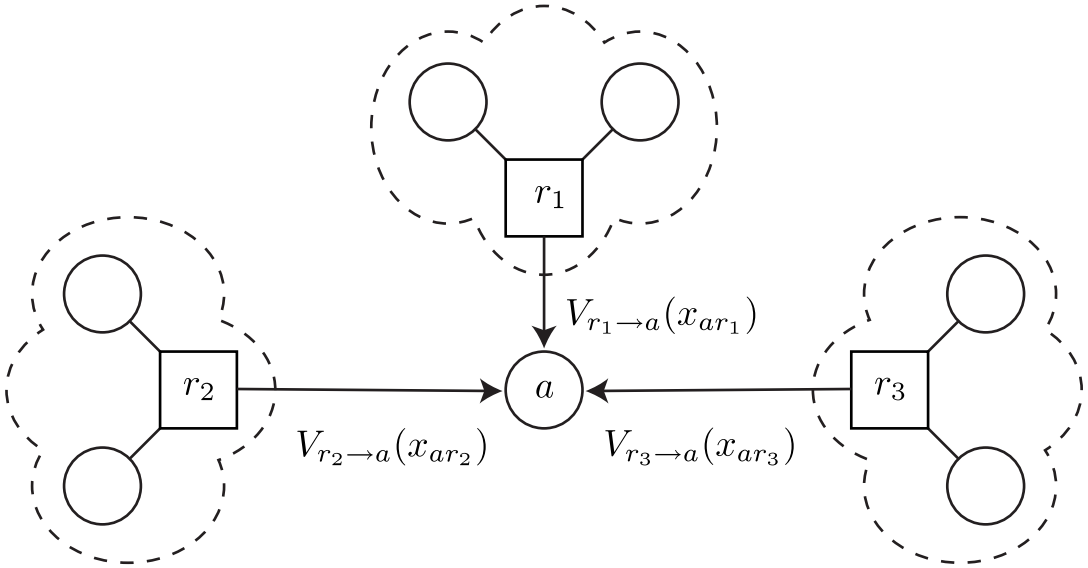


Figure 3.2 A dependency graph that is a tree. The externalities of consumption decisions for activity a decompose into three independent subproblems.

Comparing the incentives provided by the messages in (3.5) to those provided by the price-based incentives in (3.4), it is clear that messages generalize prices by allowing for nonlinear incentives. Further, with prices, there is a single price associated with each resource. Hence, the incentives corresponding to a single resource are identical to all the activities that require the resource. Messages provide additional flexibility by allowing these incentives to vary depending on the identity of the activity. This feature distinguishes message-based incentives from other work that considers nonlinear prices as proxies for decentralization (e.g., [65]).

3.4.1 Message-Passing Equilibrium

Our solution concept requires that messages obey a notion of equilibrium. We explain this intuitively now and subsequently provide a precise definition. Think of $V_{r \to a}(x_{ar})$ as a penalty imposed on activity a for consuming x_{ar} units of resource

r. The reason for penalizing the activity is that the resource can be profitably used by others. Interpret $V_{a' \to r}(x_{a'r})$ as the benefit generated by allocating $x_{a'r}$ units of the resource r to an alternative activity a'. One part of our equilibrium condition states that the penalty $V_{r \to a}(x_{ar})$ should be commensurate with the sum of benefits $V_{a' \to r}(x_{a'r})$ among the alternative activities $a' \in \partial r \setminus a$, assuming the remaining $b_r - x_{ar}$ units of the resource are allocated optimally among them. This is illustrated in Figure 3.3(a).

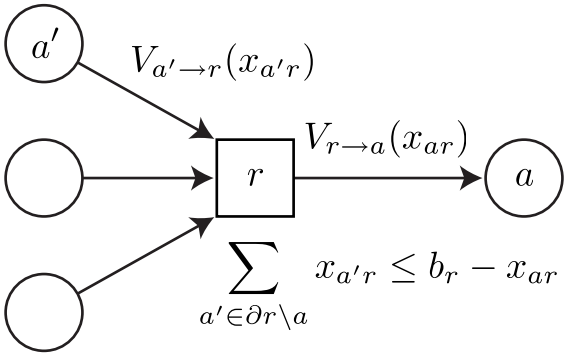
Note that, in addition to benefiting activity a, the choice of x_{ar} affects the activity's other consumption decisions $x_{ar'}$, for $r' \in \partial a \setminus r$. The benefit $V_{a \to r}(x_{ar})$ should be commensurate with the sum of the utility $u_a(x_{\partial a})$ generated by activity a and the penalties $V_{r' \to a}(x_{ar'})$ for the activity's consumption of other resources, assuming that the other resource consumption decisions are made optimally. A second equilibrium condition appropriately accounts for this cascading influence of the choice of x_{ar}. This is illustrated in Figure 3.3(b).

To define our equilibrium conditions more precisely, we introduce an operator. Denote by V an entire set of messages, including the messages from activity managers to resource managers, $\{V_{a \to r}(\cdot) \ : \forall \ a \in \mathcal{A}, \ r \in \partial a\}$, and messages from resource managers to activity managers, $\{V_{r \to a}(\cdot) \ : \ \forall \ r \in \mathcal{R}, \ a \in \partial r\}$. The operator F maps one set of messages to another and is defined by

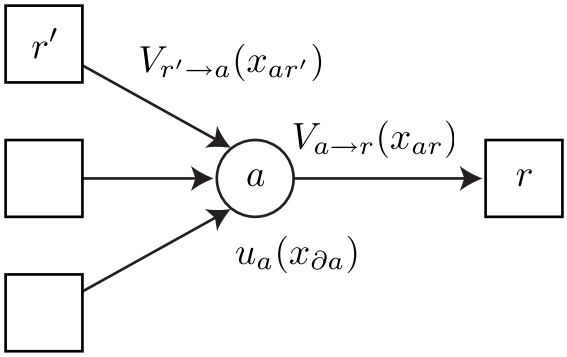
$$(FV)_{a \to r}(x_{ar}) \triangleq \begin{array}{ll} \text{maximize} & u_a(x_{\partial a}) + \sum_{r' \in \partial a \setminus r} V_{r' \to a}(x_{ar'}) \\ \text{subject to} & x_{ar'} \in \mathcal{X}_{r'}, \quad \forall \ r' \in \partial a \setminus r, \end{array} \tag{3.7a}$$

$$(FV)_{r \to a}(x_{ar}) \triangleq \begin{array}{ll} \text{maximize} & \sum_{a' \in \partial r \setminus a} V_{a' \to r}(x_{a'r}) \\ \text{subject to} & \sum_{a' \in \partial r \setminus a} x_{a'r} \leq b_r - x_{ar}, \\ & x_{a'r} \in \mathcal{X}_r, \quad \forall \ a' \in \partial r \setminus a. \end{array} \tag{3.7b}$$

The first part of the definition (3.7a) relates the benefit of allocating resource r to activity a to the penalties associated with other resource constraints associated with the activity. The second part of the definition (3.7b) relates the penalty imposed on activity a for consuming resource r to benefits that other activities could obtain.



(a) A message from a resource to an activity.



(b) A message from an activity to a resource.

Figure 3.3 The equilibrium condition for messages.

In order to better understand the operator F, consider the case where the dependency graph $\mathcal{D}$ is singly-connected, and a set of messages V satisfies the fixed point equation $V = FV$. In this case, the messages V correspond to a dynamic programming decomposition of the decision externalities for all the activities, and the operator F is analogous to a Bellman operator.

In the case where the dependency graph has cycles, the operator F may not have any fixed points. This can be addressed with a minor modification: note that adding or subtracting a constant from any message does not influence incentives. Only the relative values of a message matter. As such, we restrict attention to

messages that assign zero value to a null allocation. In other words, for each activity a and $r \in \partial a$, we consider only messages for which $V_{a\to r}(0) = 0$ and $V_{r\to a}(0) = 0$. We introduce a modified version H of the operator F which subtracts an offset[2] to accomplish this:

$$(HV)_{a\to r}(x_{ar}) \triangleq (FV)_{a\to r}(x_{ar}) - (FV)_{a\to r}(0),$$
$$(HV)_{r\to a}(x_{ar}) \triangleq (FV)_{r\to a}(x_{ar}) - (FV)_{r\to a}(0).$$

We call a set of messages V a *message-passing equilibrium* if $V = HV$. The following result, whose proof is provided in Section 3.7.1, offers a general sufficient condition for existence.

Theorem 3.1 *Assume that the utility functions are Lipschitz continuous. Then, a message-passing equilibrium exists.*

The message-passing operator H is a variation of the min-sum algorithm, as described in Chapter 2. Both methods are derived based on the assumption that the system decomposes into a collection of individual decision making agents, and that the problem faced by each agent decomposes into independent subproblems. In Chapter 2, however, we assumed that the agents corresponded to decision variables and factors of the optimization problem. In the context of resource allocation, it is more natural to assume that the agents correspond to individual activities and resources.

3.4.2 Optimality

Given a message-passing equilibrium V, an allocation can be selected by optimizing, for each activity a, the activity manager's problem (3.5). In this section, we characterize the optimality properties of this allocation.

Consider two feasible allocations x and x'. We can interpret the difference $x - x'$ as a set of *direct transfers* of resources between various activity managers

[2]The subtraction of an offset is analogous to the modification of the Bellman operator in average cost dynamic programming necessary when moving from a finite horizon to an infinite horizon setting.

and resource managers. These transfers involve pairs of activities and resources that are indexed by the set

$$\Delta(x, x') \triangleq \{(a, r) \in \mathcal{A} \times \mathcal{R} : \ x_{ar} \neq x'_{ar}\},$$

which is the collection of decision variables that differ between the two allocations.

Given an allocation x, we say that a set of direct transfers is feasible if the allocation resulting from the combination of transfers is feasible. A *cycle* is a set of transfers for which the activity-resource pairs involved can be written as

$$(a_1, r_1), (a_2, r_1), (a_2, r_2), (a_3, r_2), \ldots, (a_k, r_k), (a_1, r_k),$$

where each resource index and each activity index are distinct. The following theorem characterizes a set of transfers that cannot improve a solution delivered by a message-passing equilibrium. The proof is omitted, but can easily be established using the methods of [32, 89],

Theorem 3.2 *Given a message-passing equilibrium V, assume that each activity manager's problem* (3.5) *has a unique solution, and define x^* to be the resulting allocation. The objective value of this allocation cannot be increased by any set of transfers that involves at most one cycle.*

Theorem 3.2 guarantees, for example, that the objective cannot be improved by transfers involving redistribution of only a single resource, as such transfers contain no cycles. If the original dependency graph $\mathcal{D}$ contains at most one cycle, then *any* set of transfers contains at most one cycle. Hence, x^* is a global optimum. We comment further on this optimality property in Section 3.4.4.

3.4.3 Concave Utility Functions

In this section, we analyze message-passing equilibria in a convex resource allocation setting: we assume that the utility functions are Lipschitz continuous and strictly concave. Under this assumption, the system manager's problem (3.1) has a unique globally optimal allocation. Further, by the classical theory of Lagrange

multipliers, a supporting price vector exists. We demonstrate that the message-passing approach yields equivalent results.

To begin, note that, without loss of generality, we can restrict ourselves to message sets with concave messages, by the following analog of Theorem 3.1. The proof of this theorem is provided in Section 3.7.1.

Theorem 3.3 *There exists a message-passing equilibrium with concave and Lipschitz continuous messages.*

Given a message-passing equilibrium with concave messages, each activity manager's problem (3.5) has a strictly concave objective and a convex constraint set, and, hence, a unique optimal solution. Further, in this case, Theorem 3.2 can be strengthened to a global optimality guarantee, by the following theorem. The proof of this theorem is provided in Section 3.7.2.

Theorem 3.4 *Consider a message-passing equilibrium with concave and Lipschitz continuous messages. The resulting allocation of resources is globally optimal for the system manager's problem* (3.1)*.*

Now, let x^* be the globally optimal allocation, and assume that the system manager's objective $U(\cdot)$ is differentiable at x^*. Let $p^* \in \mathbb{R}_+^{\mathcal{R}}$ be the unique supporting price vector. For an activity a and resource $r \in \partial a$, we can think of p_r^* as a marginal incentive for the manager of activity a, in the sense that

$$\frac{\partial}{\partial x_{ar}} u_a(x_{\partial a}^*) = p_r^*.$$

We interpret this statement as saying that the marginal change in utility of deviating from the allocation x_{ar}^* is balanced by the incremental resource cost due to the price. The following theorem shows that the derivatives of messages in message-passing equilibrium can be interpreted the same way. The proof of this theorem is provided in Section 3.7.2.

Theorem 3.5 *Let x^* be the globally optimal allocation for the system manager's problem* (3.1) *and let p^* be a supporting price vector. Suppose that $U(\cdot)$ is differentiable at x^*. Consider a message-passing equilibrium V with concave and Lipschitz*

continuous messages. Then, for each activity a and resource r,

$$\frac{d}{dx_{ar}} V_{a\to r}(x^*_{ar}) = p^*_r, \quad \frac{d}{dx_{ar}} V_{r\to a}(x^*_{ar}) = -p^*_r,$$

where the existence of the above derivatives is guaranteed. Thus,

$$\frac{\partial}{\partial x_{ar}} u_a(x^*_{\partial a}) = \frac{d}{dx_{ar}} V_{a\to r}(x^*_{ar}) = -\frac{d}{dx_{ar}} V_{r\to a}(x^*_{ar}) = p^*_r.$$

Theorem 3.5 implies that, subject to differentiability considerations, concave message-passing equilibria are unique in their first-order behavior at the optimal allocation, and this behavior corresponds to that of the unique shadow price vector.

3.4.4 Messages Versus Prices

As we have discussed, shadow prices and message-passing equilibria provide two different ways to decompose the system manager's problem (3.1) into a series of smaller problems, one for each activity manager, which are of the form (3.4) and (3.5), respectively. These activity managers' problems are no longer coupled by resource constraints and can be solved independently. Both methods can be interpreted as providing incentives to each activity manager which capture decision externalities.

In the convex resource allocation case, the discussion in Section 3.4.3 suggests that these methods are equivalent. Both methods derive incentives that support the globally optimal allocation, and these incentives have equivalent structure in a local neighborhood of the globally optimal allocation.

For nonconvex problems, however, message-passing equilibria still exist and the allocations they suggest satisfy certain optimality properties. In particular, allocations derived from message-passing equilibria satisfy the optimality property of Theorem 3.2.

We can clearly contrast these methods in the case of smooth (but not concave) utilities. There, subject to regularity conditions, a price vector exists supporting every locally optimal allocation and guarantees that the objective cannot be

improved by small deviations from the prescribed allocation. A generalization of Theorem 3.4 would make a similar local optimality guarantee for a message-passing equilibrium. However, the objective also cannot be improved through very large scale deviations which satisfy the conditions of Theorem 3.2.

Finally, it should be noted that the optimality properties of message-passing equilibria are not well understood from a theoretical perspective. Their performance on many difficult optimization problems is far better than suggested by the guarantee provided by Theorem 3.2. We shall see an example of this in Section 3.6.1.

3.5 Message-Passing Algorithms

Up to this point, we have described message-passing equilibrium as a solution concept and analyzed its properties. In this section, we consider the issue of computing message-passing equilibrium.

Since a message-passing equilibrium is a fixed point of the operator H, a natural approach to consider is the method of successive approximations. This is an iterative scheme which starts with some initial message set $V^{(0)}$, for example $V^{(0)} \triangleq 0$, and generates subsequent approximations to a message-passing equilibrium according to

$$V^{(t+1)} \triangleq (1-\gamma)V^{(t)} + \gamma H V^{(t)}. \tag{3.8}$$

Here, the scalar $\gamma \in (0, 1]$ is a dampening factor. This procedure is repeated until it converges and a fixed point is reached. We generically call a successive approximation scheme of the form (3.8) a message-passing algorithm.

3.5.1 Tractability

Each iteration of the successive approximations method requires the solution of optimization problems of the following form

$$\begin{array}{ll} \text{maximize} & u_a(x_{\partial a}) + \sum_{r' \in \partial a \setminus r} V_{r' \to a}(x_{ar'}), \\ \text{subject to} & x_{ar'} \in \mathcal{X}_{r'}, \quad \forall\, r' \in \partial a \setminus r, \end{array} \tag{3.9a}$$

$$\begin{array}{ll} \text{maximize} & \sum_{a' \in \partial r \setminus a} V_{a' \to r}(x_{a'r}), \\ \text{subject to} & \sum_{a' \in \partial r \setminus a} x_{a'r} \leq b_r - x_{ar}, \\ & x_{a'r} \in \mathcal{X}_r, \quad \forall\, a' \in \partial r \setminus a. \end{array} \tag{3.9b}$$

for each activity a, resource r, and $x_{ar} \in \mathcal{X}_j$. Implicit in the application of this method is the assumption that these optimization problems can be solved efficiently.

There is special structure in the problems (3.9a) and (3.9b) that can be exploited for efficient solution by dynamic programming. Note the problem (3.9b) involves optimizing a separable objective function with an interval constraint on each variable and a single linear constraint on the sum of variable values. Such a problem can be efficiently solved as a series of one-dimensional optimization problems via dynamic programming. In many relevant cases, problem (3.9a) can similarly be decomposed into one-dimensional optimization problems. This is true, for example, if the utility function is additively separable or if utility depends only on the sum or minimum of allocated resources, as we shall see in Section 3.6.1.

3.5.2 Distributed and Asynchronous Implementation

One important characteristic of the message-passing iteration (3.8) is that it naturally lends itself to a distributed and asynchronous implementation. Imagine an implementation where the activity and resource managers operate completely independently. Consider this from the perspective of an activity manager. At each point in time, the activity manager keeps track of the most recent message it has received from each neighboring resource manager in the dependency graph. Occasionally, the activity manager can decide to send a new message to each neighboring

resource manager, based on the most recent messages it has received from other resource managers. Resource managers behave in an analogous fashion. So long as every pair of activity and resource managers communicate sufficiently often, a fixed point of this distributed and asynchronous procedure is a message-passing equilibrium. Moreover, each manager only requires knowledge of and communication with neighboring managers in the dependency graph.

In general, messages are functions over a continuous domain. As such, the algorithm, as we have formulated it, cannot be implemented on digital computers. In some cases, such as the example we consider in Section 3.6.1, the messages lie in a finite dimensional space that is closed under the message-passing operator H. In such cases, messages can be transmitted by sending a finite vector of real numbers. In the more general case, it is necessary to approximate messages using finitely parameterized representations. For example, each message can be computed at a finite number of points in its domain including the end points of the interval, and values of the message between each pair of consecutive points can be approximated by linear interpolation. This is analogous to the situation in approximate dynamic programming, where the value function is approximated using a finite parameter set. Such approximations are an active area of research, see Section 4.4 for related work.

3.5.3 Convergence

An immediate question is whether the message-passing algorithm (3.8) converges to a message-passing equilibrium. Under the hypotheses of Theorems 3.1 and 3.3, the operator H is continuous and compact. Hence, any sequence of iterates generated by successive approximation has limit points. However, these limit points may not, in general, be fixed points and thus equilibria—they may be contained in some invariant collection of message sets and may be, for example, periodically oscillating.

The question of convergence is, unfortunately, not well understood in general. If the dependency graph contains no cycles, message-passing can be seen to converge

in a finite number of iterations by simple dynamic programming arguments. Abstract conditions for convergence have been developed [86], but these are difficult to verify in specific problem instances. Convergence has also been established for certain special classes of optimization problems, such as maximum-weight matching [9], and for certain random ensembles of optimization problems [80].

One case that is well-understood, however, is when message-passing is applied to the optimization of quadratic programs. Here, we and others [90, 79, 60, 52] have established convergence so long as the objective decomposes a particular way. Moreover, this convergence continues to hold in a distributed and asynchronous setting. In one such case, which we describe in Chapter 5, a rate of convergence analysis is also available.

We have recently extended these convergence results to the optimization of unconstrained convex programs. We describe this work in Chapter 4. The main sufficient condition identified for convergence is that the objective function satisfy a certain scaled diagonal dominance condition. Unfortunately, this analysis does not apply to the resource allocation problems considered here. However, in the following section, we see that message-passing can still offer excellent solutions in the absence of convergence guarantees.

3.6 Network Rate Control

One feature of the message-passing algorithms described in the previous section is that they can be implemented in a distributed manner. This can be crucial in systems where information or computational resources are decentralized. In this section, we discuss an example involving transmission rate control in a communication network.

We consider the following model. There is a set $\mathcal{R}$ of resources, each representing a link in a communication network. Each link r has a finite capacity $b_r > 0$. There is a set $\mathcal{A}$ of activities, each representing a user who wishes to transmit data across the network. Each user a transmits data along a fixed route consisting of

the set of links $\partial a \subseteq \mathcal{R}$. This is illustrated in Figure 3.4. If the user is allocated capacity $x_{\partial a}$ along these links, it can transmit at the rate $\min_{r \in \partial a} x_{ar}$, and its utility is a function of this rate,

$$u_a(x_{\partial a}) \triangleq \tilde{u}\left(\min_{r \in \partial a} x_{ar}\right).$$

Here, we assume that the single-variable utility function $\tilde{u} : \mathbb{R}_+ \to \mathbb{R}_+$ is nondecreasing. The objective is to allocate capacity in a way that maximizes the sum of utilities.

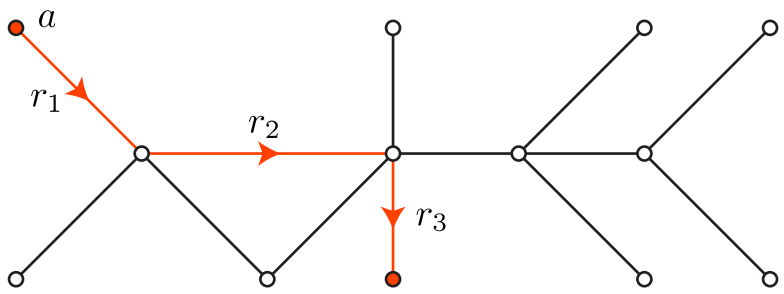


Figure 3.4 A network rate control example. Each edge in the graph is a constrained communications link. Each user is associated with a route in the network. For example, user a wishes to transmit data along the path consisting of the links $\partial a = \{r_1, r_2, r_3\}$.

The numbers of users and links in modern communication networks are enormous. As such, it is not possible for a central authority to gather all the utility functions and link capacities as would be required to make centralized allocation decisions. Rather, the capacity of each link must be allocated based on locally available information. This information should be gathered from packets of data transmitted by users as they pass through the link. Further, links might mark the packets as they pass through to inform users of how much capacity they are allocated.

For the case of increasing strictly concave utility functions, referred to in the networking literature as the case of elastic traffic [81], Kelly proposes an elegant

distributed algorithm [40]. Our interest here is in designing a distributed message-passing scheme that effectively optimizes the allocation when utility functions are not concave, also known as the case of inelastic traffic. Such utility functions are required to model user preferences, for example, in real-time video and audio applications [81]. Optimization algorithms designed for elastic traffic, like that of Kelly, can lead to instabilities when applied in the presence of inelastic traffic [45]. Several heuristics have been proposed to address inelastic traffic [45, 36, 29].

3.6.1 Inelastic Rate Control

Consider the extreme case of inelastic traffic described in Example 3.3. Here, each user a has a utility function

$$\tilde{u}_a(x_a) \triangleq z_a \mathbb{I}_{\{x_a \geq w_a\}}.$$

The quantity $w_a > 0$ is the minimum overall transmission rate desired by the user, and the quantity $z_a > 0$ is the utility derived if allocated a rate w_a or larger.

In this setting, each user a is indifferent between transmitting at rate 0 or at any rate in the interval $(0, w_a)$, and is similarly indifferent between transmitting at rate w_a and at any rate larger than w_a. Hence, the system manager's problem (3.1) is equivalent to the 0–1 integer program

$$\begin{array}{lll} \text{maximize} & \sum_{a \in \mathcal{A}} z_a y_a & \\ \text{subject to} & \sum_{a \in \partial r} w_a y_a \leq b_r, & \forall\ r \in \mathcal{R} \\ & y_a \in \{0,1\}, & \forall\ a \in \mathcal{A}. \end{array} \tag{3.10}$$

This is a multidimensional 0–1 knapsack problem, which is NP-hard [33].

There are a number of heuristics available for solving the program (3.10). A class of greedy heuristics first defines a measure of efficiency, or "bang-per-buck", for each user, representing some estimate of the contribute of the user to the overall utility relative to the cost of resource consumption of the user. We consider a prototypical efficient metric,

$$e_a \triangleq \frac{z_a}{\sum_{r \in \partial a} w_a / b_r},$$

for each user a. The users are then considered in order of decreasing efficiency, and are greedily allocated their desired capacity so long as feasibility is maintained.

Alternatively, one may consider the linear programming relaxation of (3.10). This is equivalent to approximating the utility function of each user a by the piecewise linear function

$$\tilde{u}_a(x_a) \approx \min\left(z_a, z_a x_a / w_a\right).$$

Naive application of such an approximation leads to poor solutions, as there may be many users who are allocated fractional capacities. Much better solutions can be generated from this approximation by examining the resulting vector p of shadow prices for the link constraints. These can be used as proxies for the cost of capacity on a link in order to define the efficiency metric

$$e_a \triangleq \frac{z_a}{\sum_{r \in \partial a} w_a p_r},$$

for each user a. An allocation decision can then be made as in the case of the greedy heuristic. We call this method *concave approximation*.

3.6.2 Distributed Message-Passing

Consider a distributed message-passing algorithm for a network with inelastic traffic. Here, Since the messages $V_{r\to a}(x_{ar})$ and $V_{a\to r}(x_{ar})$ represent incentives, only their values at $x_{ar} \in \{0, w_a\}$ matter. Hence, we can parameterize these messages by

$$V_{a\to r}(x_{ar}) \triangleq v_{a\to r}\mathbb{I}_{\{x_{ar}\geq w_a\}}, \qquad V_{r\to a}(x_{ar}) \triangleq v_{r\to a}\mathbb{I}_{\{x_{ar}\geq w_a\}},$$

given parameters $v_{a\to r} \geq 0$ and $v_{r\to a} \leq 0$. Denote by v the set of all parameters $\{v_{a\to r}, v_{r\to a}\}$. This parametrization is closed under the operator H, and H can be

expressed directly in terms of the parameter set v by

$$(Hv)_{a\to r} = z_a + \sum_{r'\in\partial a\setminus r} v_{r'\to a}, \tag{3.11a}$$

$$\begin{aligned} (Hv)_{r\to a} \;=\; & \text{maximize} && \textstyle\sum_{a'\in\partial r\setminus a} v_{a'\to r} y_{a'r} \\ & \text{subject to} && \textstyle\sum_{a'\in\partial r\setminus a} w_{a'} y_{a'r} \le b_r - w_a, \\ & && y_{a'r}\in\{0,1\}, \quad \forall\, a'\in\partial r\setminus a. \end{aligned} \tag{3.11b}$$

Given a a set of parameters v, each activity a needs to solve the activity manager's problem (3.5). This is equivalent to selecting to consume quantities x_{ar}, for all $r\in\partial a$, by

$$x_{ar} = \begin{cases} 0 & \text{if } z_a + \sum_{r\in\partial a} v_{r\to a} \le 0, \\ w_a & \text{if } z_a + \sum_{r\in\partial a} v_{r\to a} \ge 0. \end{cases} \tag{3.12}$$

In the case where $z_a + \sum_{r\in\partial a} v_{r\to a} = 0$, the activity manager is indifferent between consuming quantities 0 and w_a. We discuss this case in further detail in Section 3.6.3.

Since the application setting here is naturally decentralized, it is important to be able to compute the message-passing update equations (3.11a)–(3.11b) and the resulting allocation (3.12) in a distributed and possibly asynchronous fashion. We describe one particularly parsimonious implementation now.

Consider a setting where, at each time t, each link r maintains a set of incoming and outgoing message parameters $\{v_{r\to a}^{(t)},\ v_{a\to r}^{(t)}\}$ for each user $a\in\partial r$. Assume that a user a transmits a data packet at time t, along the route ∂a. A single real number m_a^+ is appended to this data packet, and the user initially sets $m_a^+ \triangleq z_a$. When the packet passes through a link $r\in\partial a$, the value of m_a^+ is observed. This value is then updated by setting $m_a^+ \triangleq m_a^+ + v_{r\to a}^{(t)}$, before it is forwarded to the next link. When the packet arrives at the destination, an acknowledgment message is sent back the source, containing a single real number m_a^-. This number is initialized to $m_a^- \triangleq 0$. As it passes through a link $r\in\partial a$, it is observed, and then updated according to $m_a^- \triangleq m_a^- + v_{r\to a}^{(t)}$, until it reaches the source.

Now, at any link $r \in \partial a$ along the route, the observed values m_a^+ and m_a^- can be combined to compute

$$m_a^+ + m_a^- = z_a + \sum_{r' \in \partial a \setminus r} v_{r' \to a}^{(t)} = (Hv^{(t)})_{a \to r}.$$

Thus, the link can update its stored incoming message from user a by setting $v_{r \to a}^{(t+1)} \triangleq m_a^+ + m_a^-$. New outgoing messages $v_{r \to a'}^{(t+1)}$, for each activity $a' \in \partial r \setminus a$, can then be computed according to the update equation (3.11b). Similarly, when the user a receives the acknowledgment packet, it can compute the value

$$z_a + m_a^- = z_a + \sum_{r \in \partial a} v_{r \to a}^{(t)}.$$

Then, it can make a consumption decision via (3.12).

The spirit of this implementation is that the computation of a message-passing equilibrium and the associated allocation decisions can be accomplished with very little overhead. All communication occurs along the normal flow of network traffic, and only a single real number is appended to every data packet.

3.6.3 Constructing Solutions

Given a collection of message parameters $\{v_{r \to a}^{(t)},\ v_{a \to r}^{(t)}\}$ at time t, individual consumption decisions for each activity can be made according to (3.12). In general, however, if there are ties in (3.12), the resulting overall allocation may not be uniquely determined. Further, it is important to ensure that the resulting overall allocation is feasible. Reminiscent of the discussion in Section 2.3.4, an additional modification is required to construct an overall allocation.

The modification we employ is a greedy rounding procedure, as described in Section 3.6.1. An efficiency metric

$$e_a \triangleq z_a + \sum_{r \in \partial a} v_{r \to a}^{(t)}$$

is defined for each user a. The users are considered sequentially in order of decreasing values of e_a. Each user is then greedily allocated capacity w_a if this is feasible, and is otherwise allocated capacity 0.

Such a modification can easily be done in a decentralized fashion in a number of ways. One such way would be as follows: imagine that each user a reports its metric e_a to each link $r \in \partial a$. Each link r then considers only the users in ∂r for which it has sufficient remaining capacity. These users are sorted by their efficiency metric values, and the link reports an "OK" message to only the highest value user, where ties are broken uniformly at random. If a user receives an "OK" message from every link, it is allocated capacity, and the process repeats with only the remaining users. This can be done be done dynamically alongside the computation of messages and the normal flow of network traffic, in the spirit of Section 3.6.2.

3.6.4 Numerical Results

We compare the performance of message-passing to the heuristics described in Section 3.6.1 as well as the optimal solution across a set of random problem instances in Figure 3.5. These problem instances are described by a size parameter, which is equal to the number of users and the number of links. The assignment of users to links is made by uniformly sampling a bipartite graph of degree 10, so that each user is assigned a route along 10 links, each link is in the route of 10 users. Each link r is assigned a fixed capacity $b_r = 5$.

The utility function of each user a is generated randomly, by setting z_a to an IID exponential random variable of mean 1, and setting $w_a \triangleq z_a$. This type of utility function corresponds to a "strongly correlated" regime for the multidimensional 0–1 knapsack problem (3.10) [33]. Here, the combinatorial nature of the underlying packing problem is most apparent and the problem is thought to be most difficult.

In these simulations, message-passing is run for 1000 iterations, independent of the problem size. During each iteration t, a set of message-passing parameters $v^{(t)}$ is updated according to $v^{(t)} = (1-\gamma)v^{(t-1)} + \gamma H v^{(t-1)}$, where a dampening factor of $\gamma = 0.5$ is used. An allocation decision $x^{(t)}$ is made by the method described in Section 3.6.3. The objective value of the best allocation seen in the 1000 iterations is reported.

In Figure 3.5, we can see the performance of message-passing versus the greedy and concave approximation heuristics. The algorithms are compared across a set of problem instances of various sizes by their optimality gap to the globally optimal allocation, which is determined using a mixed integer solver.

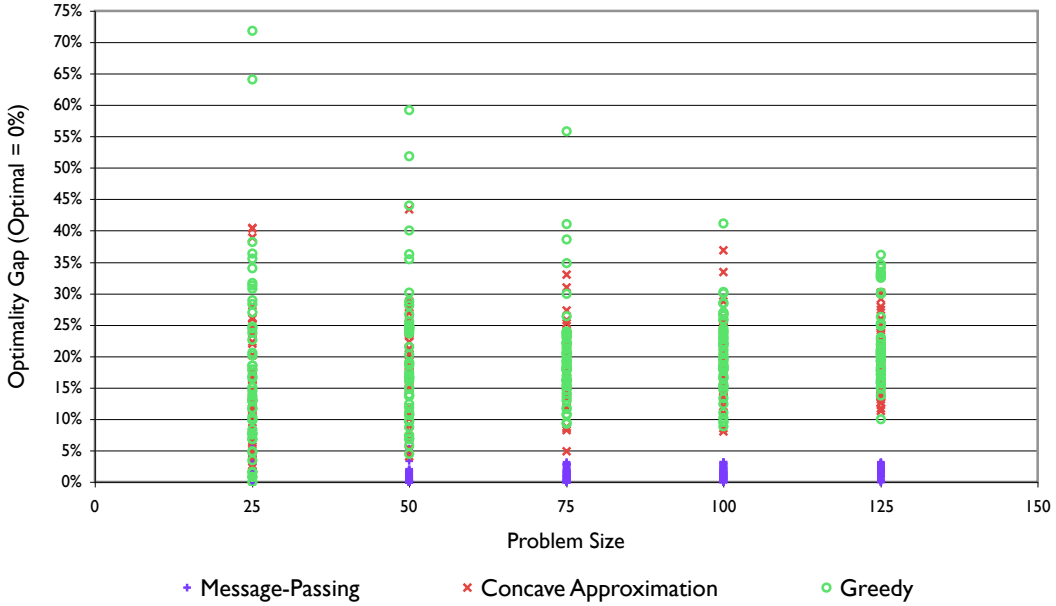


Figure 3.5 A comparison of message-passing versus competing algorithms for a set of random inelastic rate control problem instances.

Message-passing performs significantly better than either heuristic. Moreover, the optimality gap for message-passing is very consistent, and is typically within 3% of the optimal objective value. The heuristics, on the other hand, have highly variable performance across problem instances.

For this class of problems, the efficiency metric employed by the greedy algorithm is constant: $e_a = 0.5$ for each user a. Hence, the greedy heuristic is particularly trivial: consider the users in an arbitrary order, and greedily assign capacity while maintaining feasibility. The concave approximation heuristic, which requires solution of a linear program, does not perform noticeably better.

Finally, note that the problem sizes in Figure 3.6.1 are limited to 125 users. These are the largest problems for which our mixed integer solver[3] could compute a global optimum. Message-passing can comfortably scale to much larger problem instances, up to 100,000's of users on a desktop workstation. Indeed, message-passing could handle much larger problem instances than even our commercial LP solver, which was used in computing concave approximation solutions.

3.7 Proofs

In this section, we provide proofs for the main results of the chapter.

3.7.1 Proof of Theorems 3.1 and 3.3

Theorem 3.1 *Assume that the utility functions are Lipschitz continuous. Then, a message-passing equilibrium exists.*

Proof. Let L be a Lipschitz constant that applies to all utility functions. Suppose each message in the set V is Lipschitz continuous with Lipschitz constant L. Consider the message from an activity a to a resource $r \in \partial a$. Define $\mathcal{X}^{a\setminus r} \triangleq \prod_{r\in\partial a\setminus r} \mathcal{X}_r$ to be the space of consumption bundles for activity a, excluding resource r. Without loss of generality, assume that $(FV)_{a\to r}(x'_{ar}) \geq (FV)_{a\to r}(x_{ar})$. Then, for some

[3]We employed the ILOG CPLEX 9.1 mixed integer solver to compute globally optimal solutions. The LP solver from the same package was used in computing concave approximation solutions.

$z' \in \mathcal{X}^{a\setminus r}$,

$$\begin{aligned}
(FV)_{a\to r}(x'_{ar}) - (FV)_{a\to r}(x_{ar}) &= \max_{z\in\mathcal{X}^{a\setminus r}} \left(u_a(x'_{ar}, z) + \sum_{r'\in\partial a\setminus r} V_{r'\to a}(z_{ar'}) \right) \\
&\quad - \max_{z\in\mathcal{X}^{a\setminus r}} \left(u_a(x_{ar}, z) + \sum_{r'\in\partial a\setminus r} V_{r'\to a}(z_{ar'}) \right) \\
&= u_a(x'_{ar}, z') + \sum_{r'\in\partial a\setminus r} V_{r'\to a}(z'_{ar'}) \\
&\quad - \max_{z\in\mathcal{X}^{a\setminus r}} \left(u_a(x_{ar}, z) + \sum_{r'\in\partial a\setminus r} V_{r'\to a}(z_{ar'}) \right) \\
&\le u_a(x'_{ar}, z') - u_a(x_{ar}, z') \\
&\le L|x'_{ar} - x_{ar}|.
\end{aligned}$$

Hence, the message $(FV)_{a\to r}(\cdot)$ is Lipschitz continuous with Lipschitz constant L. A similar proof applies to $(FV)_{r\to a}(\cdot)$.

Let $\mathcal{S}$ be the collection of message sets V for which each message equals zero at zero and is Lipschitz continuous with Lipschitz constant L. Note that $\mathcal{S}$ is convex, closed, and bounded (under the supremum norm). $\mathcal{S}$ is a subset of the set of continuous functions from a compact, finite dimensional metric space to itself. Hence, $\mathcal{S}$ is compact under the supremum norm by the Arzelà-Ascoli theorem. The operator H maps $\mathcal{S}$ to $\mathcal{S}$ continuously with respect to the supremum norm. It follows from the Schauder fixed point theorem that a message-passing equilibrium exists. ■

Theorem 3.3 *There exists a message-passing equilibrium with concave and Lipschitz continuous messages.*

Proof. The proof follows by a modification of the proof of Theorem 3.1: define the set $\mathcal{S}'$ to be the collection of message sets $V \in \mathcal{S}$ which are also concave. Since the operator H involves maximization of a concave function over a convex set, if $V \in \mathcal{S}'$, then HV is also concave hence $HV \in \mathcal{S}'$. The existence of a fixed-point in $\mathcal{S}'$ follows from the Schauder fixed point theorem. ■

3.7.2 Proof of Theorems 3.4 and 3.5

First, consider a preliminary lemma:

Lemma 3.1 *Consider a message-passing equilibrium $HV = V$, where each activity manager's problem (3.5) has a unique solution, and denote the resulting allocation by x^*. Then, for each activity a and resource $r \in \partial a$, this allocation maximizes the optimization problems*

$$\begin{array}{lll} \textit{maximize} & T_{r\to a}(x_{\partial r}) \triangleq & \sum_{a'\in\partial r\setminus a} V_{a'\to r}(x_{a'r}) - V_{r\to a}(x_{ar}) \\ \textit{subject to} & & \sum_{a'\in\partial r} x_{a'r} \le b_r, \\ & & x_{a'r} \in \mathcal{X}_r, \qquad\qquad \forall\ a' \in \partial r, \end{array} \tag{3.13a}$$

$$\begin{array}{lll} \textit{maximize} & T_{a\to r}(x_{\partial a}) \triangleq & u_a(x_{\partial a}) + \sum_{r'\in\partial a\setminus r} V_{r'\to a}(x_{ar'}) \\ & & \quad -V_{a\to r}(x_{ar}) \\ \textit{subject to} & & x_{ar'} \in \mathcal{X}_{r'}, \qquad\qquad \forall\ r' \in \partial a. \end{array} \tag{3.13b}$$

Proof. Denote by $U_a(\cdot)$ the objective function of the activity manager's problem (3.5). From the equilibrium equation $HV = V$, and the hypothesis that $U_a(\cdot)$ has the maximizer $x^*_{\partial a}$,

$$\begin{aligned} \max_{x_{\partial a\setminus r}} U_a(x_{\partial a}) &= V_{a\to r}(x_{ar}) + V_{r\to a}(x_{ar}) + (FV)_{a\to r}(0), \\ U_a(x^*_{\partial a}) &= V_{a\to r}(x^*_{ar}) + V_{r\to a}(x^*_{ar}) + (FV)_{a\to r}(0). \end{aligned}$$

where the maximization occurs over the consumption decisions for all resources except r. Since

$$U_a(x_{\partial a}) = T_{a\to r}(x_{\partial a}) + V_{a\to r}(x_{ar}) + V_{r\to a}(x_{ar}),$$

we have

$$\begin{aligned} \max_{x_{\partial a\setminus r}} T_{a\to r}(x_{\partial a}) = (FV)_{a\to r}(0) &= U_a(x^*_{\partial a}) - V_{a\to r}(x^*_{ar}) - V_{r\to a}(x^*_{ar}) \\ &= T_{a\to r}(x^*_{\partial a}). \end{aligned}$$

Thus, $x^*_{\partial a}$ maximizes (3.13b). The result for (3.13a) is established similarly. ■

In order to prove Theorem 3.4, we first need some standard definitions from convex analysis [78]. Denote by

$$\mathcal{X} \triangleq \prod_{a \in \mathcal{A}} \prod_{r \in \partial a} \mathcal{X}_r$$

the domain of the system objective $U(\cdot)$. Given a concave function $F : \mathcal{X} \to \mathbb{R}$ which is Lipschitz continuous, define the directional derivative at a point x in the direction d by

$$\nabla_d F(x) = \lim_{\alpha \searrow 0} \frac{F(x + \alpha d) - F(x)}{\alpha}.$$

If d is such that $x + \alpha d \in \mathcal{X}$ for positive scalars α sufficiently small, the limit exists and is finite, otherwise set $\nabla_d F(x) \triangleq -\infty$. We say that $F(\cdot)$ is differentiable at x if there is a vector $\nabla F(x)$ such that, for all d, $\nabla_d F(x) = \nabla F(x) \cdot d$. Note that if $F(\cdot)$ is differentiable at x, by this definition, x must lie in the interior of $\mathcal{X}$. Right and left partial derivatives are defined through the corresponding directional derivatives by

$$\frac{\partial^+}{\partial x_{ar}} F(x) = \nabla_{e^{ar}} F(x), \quad \frac{\partial^-}{\partial x_{ar}} F(x) = -\nabla_{-e^{ar}} F(x).$$

Here, e^{ar} is a unit vector that is zero except in component ar. Denote by $\partial F(x)$ the set of supergradients to $F(\cdot)$ at x, that is

$$\partial F(x) = \{z \ : \ F(y) \leq F(x) + z \cdot (y - z), \ \forall \ y \in \mathcal{X}\}.$$

Theorem 3.4 *Consider a message-passing equilibrium with concave and Lipschitz continuous messages. The resulting allocation of resources is globally optimal for the system manager's problem* (3.1).

Proof. Consider a message-passing equilibrium V with concave and Lipschitz continuous messages, and let x^* be the associated allocation. Assume that x^* lies in the interior of the domain of $U(\cdot)$. By [78, Theorem 27.4], for each resource r and

activity a, there must exist a supergradient $d^{ar} \in \partial u_a(x^*_{\partial a})$ so that we have the first order conditions for the optimization problem (3.13b),

$$d^{ar}_{ar} - \frac{d^+}{dx_{ar}} V_{a\to r}(x^*_{ar}) \le 0, \tag{3.14a}$$

$$d^{ar}_{ar} - \frac{d^-}{dx_{ar}} V_{a\to r}(x^*_{ar}) \ge 0, \tag{3.14b}$$

$$d^{ar}_{ar'} + \frac{d^+}{dx_{ar'}} V_{a\to r'}(x^*_{ar'}) \le 0, \ \forall\ r' \in \partial a \setminus r, \tag{3.14c}$$

$$d^{ar}_{ar'} - \frac{d^-}{dx_{ar'}} V_{a\to r}(x^*_{ar'}) \ge 0, \ \forall\ r' \in \partial a \setminus r. \tag{3.14d}$$

Similarly, let $\lambda^*_{ar} \ge 0$ be a shadow price to the optimization problem (3.13a). Then,

$$-\frac{d^+}{dx_{ar}} V_{r\to a}(x^*_{ar}) - \lambda^*_{ar} \le 0, \tag{3.15a}$$

$$-\frac{d^-}{dx_{ar}} V_{r\to a}(x^*_{ar}) - \lambda^*_{ar} \ge 0, \tag{3.15b}$$

$$\frac{d^+}{dx_{a'r}} V_{a'\to r}(x^*_{a'r}) - \lambda^*_{ar} \le 0, \ \forall\ a' \in \partial r \setminus a, \tag{3.15c}$$

$$\frac{d^-}{dx_{ar}} V_{a'\to r}(x^*_{a'r}) - \lambda^*_{ar} \ge 0, \ \forall\ a' \in \partial r \setminus a. \tag{3.15d}$$

Then, by (3.14a)–(3.14b) and (3.15a)–(3.15b), and the concavity of $V_{a\to r}(\cdot)$ and $V_{r\to a}(\cdot)$,

$$\frac{d}{dx_{ar}} V_{a\to r}(x^*_{ar}) = d^{ar}_{ar}, \qquad \frac{d}{dx_{ar}} V_{r\to a}(x^*_{ar}) = -\lambda^*_{ar}. \tag{3.16}$$

By (3.15c)–(3.15d), and (3.16), we have

$$\lambda^*_{ar} = d^{a'r}_{a'r}, \quad \forall\ a' \in \partial r \setminus a.$$

Then, we must have $\lambda^*_{ar} = p^*_r$ for some vector $p^* \in \mathbb{R}^{\mathcal{R}}_+$, and, using (3.14c)–(3.14d), also

$$d^{ar}_{ar'} = p^*_{r'}, \quad \forall r' \in \partial a \setminus r.$$

Define the vector dU by $(dU)_{ar} = p_r^*$, for each $a \in \mathcal{A}$ and $r \in \partial a$. Then, $dU \in \partial U(x^*)$ is a supergradient of $U(\cdot)$ at x^*, the vector p^* is a shadow price vector for the system optimization problem (3.1), and the allocation x^* is globally optimal. The case where x^* is on the boundary of the domain of $U(\cdot)$. ■

Theorem 3.5 *Let x^* be the globally optimal allocation for the system manager's problem* (3.1) *and let p^* be a supporting price vector. Suppose that $U(\cdot)$ is differentiable at x^*. Consider a message-passing equilibrium V with concave and Lipschitz continuous messages. Then, for each activity a and resource r,*

$$\frac{d}{dx_{ar}} V_{a\to r}(x_{ar}^*) = p_r^*, \quad \frac{d}{dx_{ar}} V_{r\to a}(x_{ar}^*) = -p_r^*,$$

where the existence of the above derivatives is guaranteed. Thus,

$$\frac{\partial}{\partial x_{ar}} u_a(x_{\partial a}^*) = \frac{d}{dx_{ar}} V_{a\to r}(x_{ar}^*) = -\frac{d}{dx_{ar}} V_{r\to a}(x_{ar}^*) = p_r^*.$$

Proof. This follows by the same argument as in Theorem 3.4, and the fact that if $U(\cdot)$ is differentiable at x^*, $\partial U(x^*) = \{\nabla U(x^*)\}$. ■

4

UNCONSTRAINED CONVEX OPTIMIZATION

Consider a graphical model consisting of a hypergraph $(V, \mathcal{C})$, and an associated optimization problem

$$\begin{array}{lll} \underset{x}{\text{minimize}} & F(x) \triangleq & \sum_{C \in \mathcal{C}} f_C(x_C) \\ \text{subject to} & & x \in \mathbb{R}^V. \end{array} \tag{4.1}$$

Here, each decision variable x_i is real-valued, and we assume that each factor $f_C : \mathbb{R} \to \mathbb{R}$ is real-valued and convex. The optimization problem (4.1) is clearly convex. We will refer to such programs generically as *separable* convex programs, however, to emphasize that the factors are convex.

In this chapter, we consider the behavior of the min-sum algorithm for separable convex programs. One case that has been examined previously in the literature is where the objective is pairwise (i.e., $|C| \leq 2$, for all $C \in \mathcal{C}$) and the component functions $\{f_C(\cdot)\}$ are quadratic and convex. Here, the min-sum algorithm is known to compute the optimal solution when it converges [90, 79, 88], and sufficient conditions are known that identify a broad class of problems for which the min-sum algorithm converges [60, 52].

The main contribution of this chapter is the analysis of cases where the functions are convex but not necessarily quadratic. We establish that the min-sum algorithm and its asynchronous variants converge for a large class of such problems. The main sufficient condition is that of *scaled diagonal dominance*. This

condition is similar to known sufficient conditions for asynchronous convergence of other decentralized optimization algorithms, such as coordinate descent and gradient descent.

Analysis of the convex case has been an open challenge and its resolution advances the state of understanding in the growing literature on message-passing algorithms. Further, it builds a bridge between this emerging research area and the better established fields of convex analysis and optimization.

This chapter is organized as follows. The next section establishes a convergence result for the min-sum algorithm in the context of pairwise separable convex programs. Section 4.2 extends this result to more general separable convex programs, where each factor can be a function of more than two variables. In Section 4.3, we discuss how our convergence results hold in a totally asynchronous model of computation. When applied to a continuous optimization problem, messages computed and stored by the min-sum algorithm are functions over continuous domains. Except in very special cases, this is not feasible for digital computers, and in Section 4.4, we discuss implementable approaches to approximating the behavior of the min-sum algorithm. We close by discussing possible extensions and open issues in Section 4.5.

4.1 Pairwise Separable Convex Programs

Consider first the case of pairwise separable programs. These are programs of the form (4.1), where $|C| \leq 2$, for all $C \in \mathcal{C}$. In this case, we can define an undirected graph (V, E) based on the objective function. This graph has a vertex set V corresponding to the decision variables, and an edge set E defined by the pairwise factors, $E \triangleq \{C \in \mathcal{C} : \ |C| = 2\}$.

Definition 4.1 (Pairwise Separable Convex Program) *A pairwise separable convex program is an optimization problem of the form*

$$\begin{array}{lll} \underset{x}{\text{minimize}} & F(x) \triangleq & \sum_{i \in V} f_i(x_i) + \sum_{(i,j) \in E} f_{ij}(x_i, x_j) \\ \text{subject to} & & x \in \mathbb{R}^V, \end{array} \tag{4.2}$$

where the factors $\{f_i(\cdot)\}$ *are strictly convex, coercive, and twice continuously differentiable, the factors* $\{f_{ij}(\cdot,\cdot)\}$ *are convex and twice continuously differentiable, and*

$$M \triangleq \min_i \inf_x \frac{\partial^2}{\partial x_i^2} F(x) > 0.$$

Under this definition, the objective function $F(x)$ is strictly convex and coercive. Hence, we can define $x^* \in \mathbb{R}^V$ to be the unique optimal solution.

4.1.1 The Min-Sum Algorithm

The min-sum algorithm, as discussed in Section 2.3 for pairwise graphical models, immediately applies to pairwise separable convex programs of the form (4.2). For each vertex $i \in V$, denote the set of neighbors of i in the graph by $N(i) \triangleq \{j \in V : (i,j) \in E\}$. Denote the set of edges with direction distinguished by $\vec{E} \triangleq \{(i,j) \in V \times V : i \in N(j)\}$.

At time t, each vertex i keeps track of a message from each neighbor $u \in N(i)$. This message takes the form of a function $J_{u\to i}^{(t)} : \mathbb{R}\to\mathbb{R}$. These incoming messages are combined to compute new outgoing messages for each neighbor. The message $J_{i\to j}^{(t+1)}(\cdot)$ from vertex i to vertex $j \in N(i)$ evolves according to

$$J_{i\to j}^{(t+1)}(x_j) \triangleq \min_{x_i} \kappa_{i\to j}^{(t)} + f_i(x_i) + f_{ij}(x_i,x_j) + \sum_{u\in N(i)\setminus j} J_{u\to i}^{(t)}(x_i). \tag{4.3}$$

Here, $\kappa_{i\to j}^{(t)}$ is a normalization term.

At each time $t > 0$, a local objective function $b_j^{(t)}(\cdot)$ is defined for each variable x_j by

$$b_j^{(t)}(x_j) \triangleq f_j(x_j) + \sum_{u\in N(j)} J_{u\to j}^{(t)}(x_j). \tag{4.4}$$

An estimate $x_j^{(t)}$ can be obtained for the optimal value of the variable x_j by minimizing the local objective function:

$$x_j^{(t)} \triangleq \operatorname*{argmin}_{x_i} b_j^{(t)}(x_j). \tag{4.5}$$

The min-sum algorithm requires an initial set of messages $\{J_{i\to j}^{(0)}(\cdot)\}$ at time $t = 0$. We make the following assumption regarding these messages:

Assumption 4.1 (Min-Sum Initialization) *Assume that the each initial message $J_{i\to j}^{(0)}(\cdot)$ is twice continuously differentiable and that there exists some $z_{i\to j} \in \mathbb{R}$ with*

$$\frac{d^2}{dx_j^2} J_{i\to j}^{(0)}(x_j) \geq \frac{\partial^2}{\partial x_j^2} f_{ij}(z_{i\to j}, x_j), \quad \forall\ x_j \in \mathbb{R}. \tag{4.6}$$

Assumption 4.1 guarantees that the messages at time $t = 0$ are convex functions. Examining the update equation (4.3), it is clear that, by induction, this implies that all future messages are also convex functions. Similarly, since the functions $\{f_i(\cdot)\}$ are strictly convex and coercive, and the functions $\{f_{ij}(\cdot,\cdot)\}$ are convex, it follows that the optimization problem in the update equation (4.3) is well-defined and uniquely minimized. Finally, each local objective function $b_j^{(t)}(\cdot)$ must strictly convex and coercive, and hence each estimate $x_j^{(t)}$ is uniquely defined by (4.5).

Assumption 4.1 also requires that the initial messages be sufficiently convex, in the sense of (4.6). As we will shortly see, this will be an important condition for our convergence results. For the moment, however, note that it is easy to select a set of initial messages satisfying Assumption 4.1. For example, one might choose

$$J_{i\to j}^{(0)}(x_j) \triangleq f_{ij}(0, x_j).$$

4.1.2 Convergence

Our goal is to understand conditions under which the min-sum algorithm converges to the optimal solution x^*, i.e.

$$\lim_{t\to\infty} x^{(t)} = x^*.$$

Consider the following diagonal dominance condition:

Definition 4.2 (Scaled Diagonal Dominance) *An objective function $F: \mathbb{R}^V \to \mathbb{R}$ is (λ, w)-scaled diagonally dominant if λ is a scalar with $0 < \lambda < 1$ and $w \in \mathbb{R}^V$ is a vector with $w > 0$, so that for each $i \in V$ and all $x \in \mathbb{R}^V$,*

$$\sum_{j \in V \setminus i} w_j \left| \frac{\partial^2}{\partial x_i \partial x_j} F(x) \right| \leq \lambda w_i \frac{\partial^2}{\partial x_i^2} F(x).$$

Our main convergence result is as follows:

Theorem 4.1 *Consider a pairwise separable convex program with an objective function that is (λ, w)-scaled diagonally dominant. Assume that the min-sum algorithm is initialized in accordance with Assumption 4.1. Define the constant*

$$K \triangleq \frac{1}{M} \frac{\max_u w_u}{\min_u w_u}.$$

Then, the iterates of the min-sum algorithm satisfy

$$\|x^{(t)} - x^*\|_\infty \leq K \frac{\lambda^t}{1-\lambda} \sum_{(u,v) \in \vec{E}} \left| \frac{d}{dx_v} J^{(0)}_{u \to v}(x^*_v) - \frac{\partial}{\partial x_v} f_{uv}(x^*_u, x^*_v) \right|.$$

Hence,

$$\lim_{t \to \infty} x^{(t)} = x^*.$$

Proof. The proof for Theorem 4.1 will be provided in Section 4.1.4. ■

We can compare Theorem 4.1 to existing results on min-sum convergence in the case of where the objective function $F(\cdot)$ is quadratic. Rusmevichientong and Van Roy [79] developed abstract conditions for convergence, but these conditions are difficult to verify in practical instances. Convergence has also been established in special cases arising in certain applications [58, 63].

More closely related to our current work, Weiss and Freeman [90] established convergence when the factors $\{f_i(\cdot), f_{ij}(\cdot,\cdot)\}$ are quadratic, the single-variable factors $\{f_i(\cdot)\}$ are strictly convex, and the pairwise factors $\{f_{ij}(\cdot,\cdot)\}$ are convex and diagonally dominated, i.e.

$$\left| \frac{\partial^2}{\partial x_i \partial x_j} f_{ij}(x_i, x_j) \right| \leq \frac{\partial^2}{\partial x_i^2} f_{ij}(x_i, x_j), \quad \forall\ (i,j) \in E,\ x_j, x_j \in \mathbb{R}.$$

The results of Malioutov, et al. [52] and our prior work [60] remove the diagonal dominance assumption. However, all of these results are special cases of Theorem 4.1. In particular, if a quadratic objective function $F(\cdot)$ decomposes into pairwise factors so that the single-variable factors are quadratic and strictly convex, and the pairwise factors are quadratic convex, then $F(\cdot)$ must be scaled diagonally dominant. This can be established as a consequence of the Perron-Frobenius theorem [52]. Finally, as we will see in Section 4.2, Theorem 4.1 also generalizes beyond pairwise decompositions.

4.1.3 The Computation Tree

In order to prove Theorem 4.1, we first introduce the notion of the *computation tree*. This is a useful device in the analysis of message-passing algorithms, originally introduced by Wiberg [91]. Given a vertex $r \in V$ and a time t, the computation tree defines an optimization problem that is constructed by "unrolling" all the optimizations involved in the computation of the min-sum estimate $x_r^{(t)}$.

Formally, the computation tree is a graph $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ where each vertex $i \in \mathcal{V}$ is in labeled by a vertex $\sigma_i \in V$ in the original graph, through a mapping $\sigma : \mathcal{V} \rightarrow V$. This mapping is required to preserve the edge structure of the graph, so that if $(i, j) \in \mathcal{E}$, then $(\sigma_i, \sigma_j) \in E$. Given a vertex $i \in \mathcal{V}$, we will abuse notation and refer to the corresponding vertex $\sigma_i \in V$ in the original graph simply by i.

Fixing a vertex $r \in V$ and a time t, the computation tree rooted at r and of depth t is defined in an iterative fashion. Initially, the tree consists of single root vertex corresponding to r. At each subsequent step, the leaves in the computation tree are examined. Given a leaf i with a parent j, a vertex u and an edge (u, i) are added to the computation tree corresponding to each neighbor of i excluding j in the original graph. This process is repeated for t steps. An example of the resulting graph is illustrated in Figure 4.1.

Given the graph $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, and the correspondence mapping σ, define a decision variable x_i for each vertex $i \in \mathcal{V}$. Define a pairwise separable objective function $F_{\mathcal{T}} : \mathbb{R}^{\mathcal{V}} \rightarrow \mathbb{R}$, by considering factors of the form:

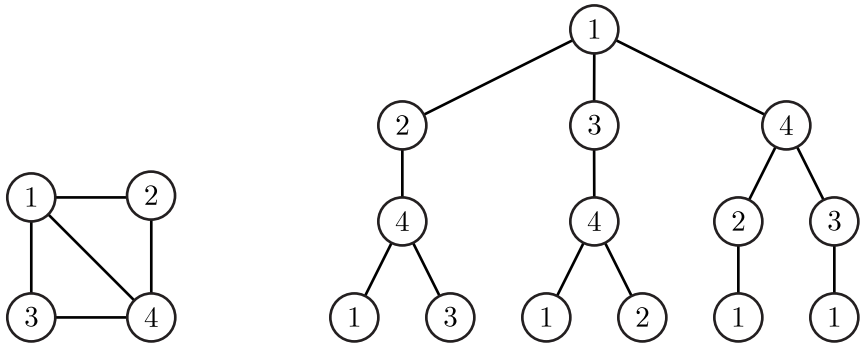


Figure 4.1 A graph and the corresponding computation tree, rooted at vertex 1 and of depth $t = 3$. The vertices in the computation tree are labeled according to the corresponding vertex in the original graph.

1. For each $i \in \mathcal{V}$, add a single-variable factor $f_i(x_i)$ by setting $f_i(x_i) \triangleq f_{\sigma_i}(x_i)$.

2. For each $(i, j) \in \mathcal{V}$, add a pairwise factor $f_{ij}(x_i, x_j)$ by setting $f_{ij}(x_i, x_j) \triangleq f_{\sigma_i \sigma_j}(x_i, x_j)$.

3. For each $i \in \mathcal{V}$ that is a leaf vertex with parent j, add a single-variable factor $J^{(0)}_{u \to \sigma_i}(x_i)$, for each neighbor $u \in N(\sigma_i) \setminus \sigma_j$ of i in the original graph, excluding j.

Now, let $\tilde{x}$ be the optimal solution to the minimization of the computation tree objective $F_{\mathcal{T}}(\cdot)$. By inductively examining the operation of the min-sum algorithm, it is easy to establish that the component $\tilde{x}_r$ of this solution at the root of the tree is precisely the min-sum estimate $x_r^{(t)}$.

The following lemma establishes that the computation tree inherits the scaled diagonal dominance property from the original objective function.

Lemma 4.1 *Consider a pairwise separable convex program with an objective function that is (λ, w)-scaled diagonally dominant. Assume that the min-sum algorithm is initialized in accordance with Assumption 4.1, and let $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ be a computation tree associated with this program. Then, the computation tree objective function $F_{\mathcal{T}}(\cdot)$ is also (λ, w)-scaled diagonally dominant.*

Proof. Given a vertex $i \in \mathcal{V}$, let $N^{\mathcal{V}}(i)$ be the neighborhood in the computation tree, and let $N(i)$ be the neighborhood of the corresponding vertex in the original graph. If $i \in \mathcal{V}$ is an interior vertex of the computation tree, then

$$\begin{aligned}
&\sum_{u \in \mathcal{V} \setminus i} w_u \left| \frac{\partial^2}{\partial x_i \partial x_u} F_{\mathcal{T}}(x) \right| \\
&= \sum_{u \in N^{\mathcal{V}}(i)} w_u \left| \frac{\partial^2}{\partial x_i \partial x_u} f_{iu}(x_i, x_u) \right| \\
&\leq \lambda w_i \left(\frac{\partial^2}{\partial x_i^2} f_i(x_i) + \sum_{u \in N^{\mathcal{V}}(i)} \frac{\partial^2}{\partial x_i^2} f_{iu}(x_i, x_u) \right) \\
&= \lambda w_i \frac{\partial^2}{\partial x_i^2} F_{\mathcal{T}}(x),
\end{aligned}$$

where the inequality follows from the scaled diagonal dominance of the original objective function $F(\cdot)$.

Similarly, if i is a leaf vertex with parent j,

$$\begin{aligned}
&\sum_{u \in \mathcal{V} \setminus i} w_u \left| \frac{\partial^2}{\partial x_i \partial x_u} F_{\mathcal{T}}(x) \right| \\
&= w_j \left| \frac{\partial^2}{\partial x_i \partial x_j} f_{ij}(x_i, x_j) \right| \\
&\leq w_j \left| \frac{\partial^2}{\partial x_i \partial x_j} f_{ij}(x_i, x_j) \right| + \sum_{u \in N(i) \setminus j} w_u \left| \frac{\partial^2}{\partial x_i \partial x_u} f_{iu}(x_i, z_{u \to i}) \right| \\
&\leq \lambda w_i \left(\frac{\partial^2}{\partial x_i^2} f_i(x_i) + \frac{\partial^2}{\partial x_i^2} f_{ij}(x_i, x_j) + \sum_{u \in N(i) \setminus j} \frac{\partial^2}{\partial x_i^2} f_{iu}(x_i, z_{u \to i}) \right) \\
&\leq \lambda w_i \left(\frac{\partial^2}{\partial x_i^2} f_i(x_i) + \frac{\partial^2}{\partial x_i^2} f_{ij}(x_i, x_j) + \sum_{u \in N(i) \setminus j} \frac{\partial^2}{\partial x_i^2} J^{(0)}_{u \to i}(x_i) \right) \\
&= \lambda w_i \frac{\partial^2}{\partial x_i^2} F_{\mathcal{T}}(x).
\end{aligned}$$

Here, the second inequality follows from the scaled diagonal dominance of the original objective function $F(\cdot)$, and the third inequality follows from Assumption 4.1. ■

4.1.4 Proof of Theorem 4.1

In order to prove Theorem 4.1, we will study the evolution of the min-sum algorithm under a set of linear perturbations. Consider an arbitrary vector $p \in \mathbb{R}^{\vec{E}}$ with one component $p_{i\to j}$ for each $i \in V$ and $j \in N(i)$. Given an arbitrary vector p, define $\{J^{(t)}_{i\to j}(\cdot, p)\}$ to be the set of messages that evolve according to

$$J^{(0)}_{i\to j}(x_j, p) \triangleq J^{(0)}_{i\to j}(x_j) + p_{i\to j}x_j, \tag{4.7}$$

$$J^{(t+1)}_{i\to j}(x_j, p) \triangleq \min_{x_i} \kappa^{(t)}_{i\to j} + f_i(x_i) + f_{ij}(x_i, x_j) + \sum_{u\in N(i)\setminus j} J^{(t)}_{u\to i}(x_i, p). \tag{4.8}$$

Similarly, define $\{b^{(t)}_j(\cdot, p)\}$ and $\{x^{(t)}_j(p)\}$ to be the resulting local objective functions and optimal value estimates under this perturbation:

$$b^{(t)}_j(x_j) \triangleq f_j(x_j) + \sum_{u\in N(j)} J^{(t)}_{u\to j}(x_j, p), \tag{4.9}$$

$$x^{(t)}_j \triangleq \operatorname*{argmin}_{x_i} b^{(t)}_j(x_j, p). \tag{4.10}$$

The following simple lemma gives a particular choice of p for which the min-sum algorithm yields the optimal solution at every time.

Lemma 4.2 *Define the vector $p^* \in \mathbb{R}^{\vec{E}}$ by setting, for each $i \in V$ and $j \in N(i)$,*

$$p^*_{i\to j} \triangleq \frac{\partial}{\partial x_j} f_{ij}(x^*_i, x^*_j) - \frac{d}{dx_j} J^{(0)}_{i\to j}(x^*_j).$$

Then, at every time $t \geq 0$,

$$\frac{\partial}{\partial x_j} J^{(t)}_{i\to j}(x^*_j, p^*) = \frac{\partial}{\partial x_j} f_{ij}(x^*_i, x^*_j), \tag{4.11}$$

and $x^{(t)}_j(p^*) = x^*_j$.

Proof. Note that the first order optimality conditions for $F(x)$ at x^* imply that, for each $j \in V$,

$$\frac{d}{dx_j} f_i(x^*_j) + \sum_{i\in N(j)} \frac{\partial}{\partial x_j} f_{ij}(x^*_i, x^*_j) = 0.$$

If (4.11) holds at time t, then this implies that

$$\frac{d}{dx_j} f_i(x_j^*) + \sum_{i \in N(j)} \frac{\partial}{\partial x_j} J_{i \to j}^{(t)}(x_j^*, p^*) = \frac{\partial}{\partial x_j} b_j^{(t)}(x_j^*, p^*) = 0.$$

This is exactly the first order optimality condition for the minimization of $b_j^{(t)}(\cdot, p^*)$, thus $x_j^{(t)}(p^*) = x_j^*$.

Clearly (4.11) holds at time $t = 0$. Assume it holds at time $t \geq 0$. Then, when $x_j = x_j^*$, the minimizing value of x_i in (4.8) is x_i^*. Hence, (4.11) holds at time $t + 1$. ■

Next, we will bound the sensitivity of the estimate $x_i^{(t)}(p)$ to the choice of p. The main technique employed here is analysis of the computation tree described in Section 4.1.3. In particular, the perturbation p impacts the computation tree only through the leaf vertices at depth t. The scaled diagonal dominance property of the computation tree, provided by Lemma 4.1, can then be used to guarantee that this impact is diminishing in t.

Lemma 4.3 *We have, for all $p \in \mathbb{R}^{\vec{E}}$, $r \in V$, $(u, v) \in \vec{E}$, and $t \geq 0$,*

$$\left| \frac{\partial}{\partial p_{u \to v}} x_r^{(t)}(p) \right| \leq K \frac{\lambda^t}{1 - \lambda}.$$

Proof. Fix $r \in V$, and let $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ be the computation tree rooted at r after t time steps. Let $F_{\mathcal{T}}(x, p)$ be the objective value of this computation tree, and let

$$\tilde{x}(p) \triangleq \operatorname*{argmin}_x \; F_{\mathcal{T}}(x, p),$$

so that

$$\tilde{x}_r(p) = x_r^{(t)}(p).$$

By the first order optimality conditions, for any $j \in \mathcal{V}$,

$$\frac{\partial}{\partial x_j} F_{\mathcal{T}}(\tilde{x}(p), p) = 0.$$

If j is an interior vertex of $\mathcal{T}$, this becomes

$$\frac{d}{dx_j} f_j(\tilde{x}_j(p)) + \sum_{i \in N(j)} \frac{\partial}{\partial x_j} f_{ij}(\tilde{x}_i(p), \tilde{x}_j(p)) = 0. \tag{4.12}$$

If j is a leaf with parent u, we have

$$\frac{d}{dx_j} f_j(\tilde{x}_j(p)) + \frac{\partial}{\partial x_j} f_{uj}(\tilde{x}_u(p), \tilde{x}_j(p)) + \sum_{i \in N(j) \setminus u} \left(\frac{\partial}{\partial x_j} J^{(0)}_{i \to j}(\tilde{x}_j(p)) + p_{i \to j} \right) = 0. \tag{4.13}$$

Now, fix some directed edge (a, b), and differentiate (4.12)–(4.13) with respect to $p_{a \to b}$. We have, for an interior vertex j,

$$\begin{aligned} 0 = {} & \frac{d^2}{dx_j^2} f_j(\tilde{x}_j(p)) \frac{\partial}{\partial p_{a \to b}} \tilde{x}_j(p) \\ & + \sum_{i \in N(j)} \frac{\partial^2}{\partial x_j^2} f_{ij}(\tilde{x}_i(p), \tilde{x}_j(p)) \frac{\partial}{\partial p_{a \to b}} \tilde{x}_j(p) \\ & + \sum_{i \in N(j)} \frac{\partial^2}{\partial x_i \partial x_j} f_{ij}(\tilde{x}_i(p), \tilde{x}_j(p)) \frac{\partial}{\partial p_{a \to b}} \tilde{x}_i(p), \end{aligned}$$

and for a leaf vertex j with parent u,

$$\begin{aligned} 0 = {} & \frac{d^2}{dx_j^2} f_j(\tilde{x}_j(p)) \frac{\partial}{\partial p_{a \to b}} \tilde{x}_j(p) \\ & + \frac{\partial^2}{\partial x_j^2} f_{uj}(\tilde{x}_u(p), \tilde{x}_j(p)) \frac{\partial}{\partial p_{a \to b}} \tilde{x}_j(p) \\ & + \frac{\partial^2}{\partial x_u \partial x_j} f_{uj}(\tilde{x}_u(p), \tilde{x}_j(p)) \frac{\partial}{\partial p_{a \to b}} \tilde{x}_u(p) \\ & + \sum_{i \in N(j) \setminus u} \left(\frac{\partial^2}{\partial x_j^2} J^{(0)}_{i \to j}(\tilde{x}_j(p)) \frac{\partial}{\partial p_{a \to b}} \tilde{x}_j(p) + \mathbb{I}_{\{(a,b)=(i,j)\}} \right). \end{aligned}$$

We can write this system of equations in matrix form, as

$$\Gamma v^{a \to b} + h^{a \to b} = 0. \tag{4.14}$$

Here, $v^{a \to b} \in \mathbb{R}^{\mathcal{V}}$ is a vector with components

$$v_j^{a \to b} \triangleq \frac{\partial}{\partial p_{a \to b}} \tilde{x}_j(p).$$

The vector $h^{a \to b} \in \mathbb{R}^{\mathcal{V}}$ has components

$$h_j^{a \to b} \triangleq \mathbb{I}_{\{j \text{ is a leaf vertex of type } a \text{ with a parent of type } b\}}.$$

The symmetric matrix $\Gamma \in \mathbb{R}^{\mathcal{V} \times \mathcal{V}}$ has components as follows:

1. If j is an interior vertex,

$$\Gamma_{jj} \triangleq \frac{d^2}{dx_j^2} f_j(\tilde{x}_j(p)) + \sum_{i \in N(j)} \frac{\partial^2}{\partial x_j^2} f_{ij}(\tilde{x}_i(p), \tilde{x}_j(p)).$$

2. If j is an interior vertex and $i \in N(j)$,

$$\Gamma_{ij} \triangleq \frac{\partial^2}{\partial x_i \partial x_j} f_{ij}(\tilde{x}_i(p), \tilde{x}_j(p)).$$

3. If j is a leaf vertex with parent u,

$$\begin{aligned}\Gamma_{jj} &\triangleq \frac{d^2}{dx_j^2} f_j(\tilde{x}_j(p)) + \frac{\partial^2}{\partial x_j^2} f_{uj}(\tilde{x}_u(p), \tilde{x}_j(p)) \\ &\quad + \sum_{i \in N(j) \setminus u} \frac{\partial^2}{\partial x_j^2} J_{i \to j}^{(0)}(\tilde{x}_j(p)), \\ \Gamma_{uj} &\triangleq \frac{\partial^2}{\partial x_u \partial x_j} f_{uj}(\tilde{x}_u(p), \tilde{x}_j(p)).\end{aligned}$$

4. All other entries of Γ are zero.

Note that $\Gamma = \nabla_x^2 F_{\mathcal{T}}(\tilde{x}(p), p)$. Then, Lemma 4.1 implies that

$$\sum_{i \in \mathcal{V} \setminus j} w_i |\Gamma_{ij}| \leq \lambda w_j \Gamma_{jj}. \tag{4.15}$$

Define, for vectors $x \in \mathbb{R}^{\mathcal{V}}$, the weighted sup-norm

$$\|x\|_\infty^w \triangleq \max_j \ |x_j|/w_j.$$

For a linear operator $A : \ \mathbb{R}^{\mathcal{V}} \to \mathbb{R}^{\mathcal{V}}$, the corresponding induced operator norm is given by

$$\|A\|_\infty^w \triangleq \max_j \ \frac{1}{w_j} \sum_{i \in \mathcal{V}} w_i |A_{ji}|.$$

Define the matrices

$$\begin{aligned}D &\triangleq \operatorname{diag}(\Gamma), \\ R &\triangleq I - D^{-1}\Gamma.\end{aligned}$$

Then, (4.15) implies that

$$\|R\|_\infty^w \leq \lambda < 1.$$

Hence, the matrix $I - R = D^{-1}\Gamma$ is invertible, and

$$\left(D^{-1}\Gamma\right)^{-1} = (I-R)^{-1} = \sum_{s=0}^{\infty} R^s.$$

Examining the linear equation (4.14), we have

$$v^{a\to b} = -\Gamma^{-1}h^{a\to b} = -(I-R)^{-1}D^{-1}h^{a\to b} = -\sum_{s=0}^{\infty} R^s D^{-1}h^{a\to b}.$$

We are interested in bounding the value of the component $v_r^{a\to b}$ (recall that $v_r^{a\to b} = \partial x_r^{(t)}(p)/\partial p_{a\to b}$). Hence, we have

$$v_r^{a\to b} = -\sum_{s=0}^{\infty}\left[R^s D^{-1}h^{a\to b}\right]_r.$$

Since $h^{a\to b}$ is zero on interior vertices, and any leaf vertex is distance t from the root r, we have

$$\left[R^s D^{-1}h^{a\to b}\right]_r = 0, \quad \forall\ s < t.$$

Thus,

$$v_r^{a\to b} = -\sum_{s=t}^{\infty}\left[R^s D^{-1}h^{a\to b}\right]_r.$$

Then,

$$\begin{aligned}
|v_r^{a\to b}|/w_r &\leq \left\|\sum_{s=t}^{\infty} R^s D^{-1}h^{a\to b}\right\|_\infty^w \\
&\leq \sum_{s=t}^{\infty}\|R^s\|_\infty^w \left\|D^{-1}h^{a\to b}\right\|_\infty^w \\
&\leq \frac{\lambda^t}{1-\lambda}\left\|D^{-1}h^{a\to b}\right\|_\infty^w \\
&\leq \frac{\lambda^t}{1-\lambda}\max_{i\in V}\sup_{x\in\mathbb{R}^V}\left(w_i\frac{\partial^2}{\partial x_i^2}F(x)\right)^{-1} \\
&\leq M\frac{\lambda^t}{1-\lambda}\max_{i\in V}\frac{1}{w_i}.
\end{aligned}$$

■

The following lemma combines the results from Lemmas 4.2 and 4.3. Theorem 4.1 follows by taking $p = 0$.

Lemma 4.4 *Given an arbitrary vector $p \in \mathbb{R}^{\vec{E}}$,*

$$\|x^{(t)}(p) - x^*\|_\infty \leq K \frac{\lambda^t}{1-\lambda} \sum_{(u,v)\in\vec{E}} |p_{u\to v} - p^*_{u\to v}| \,.$$

Proof. For any $j \in V$, define

$$g_j^{(t)}(\theta) = x_j^{(t)}(\theta p + (1-\theta)p^*).$$

We have, from Lemma 4.2,

$$x_j^{(t)}(p) - x_j^* = x_j^{(t)}(p) - x_j^{(t)}(p^*) = g_j^{(t)}(1) - g_j^{(t)}(0).$$

By the mean value theorem and Lemma 4.3,

$$\begin{aligned}
&|x_j^{(t)}(p) - x_j^*| \\
&\leq \sup_{\theta\in[0,1]} \left| \frac{d}{d\theta} g_j^{(t)}(\theta) \right| \\
&\leq \sup_{\theta\in[0,1]} \sum_{(u,v)\in\vec{E}} \left| \frac{\partial}{\partial p_{u\to v}} x_j^{(t)}(\theta p + (1-\theta)p^*) \right| |p_{u\to v} - p^*_{u\to v}| \\
&\leq K \frac{\lambda^t}{1-\lambda} \sum_{(u,v)\in\vec{E}} |p_{u\to v} - p^*_{u\to v}|.
\end{aligned}$$

■

4.2 General Separable Convex Programs

In this section we will consider convergence of the min-sum algorithm for more general separable convex programs. In particular, consider a vector of real-valued decision variables $x \in \mathbb{R}^V$, indexed by a finite set V, and a hypergraph $(V, \mathcal{C})$, where the set $\mathcal{C}$ is a collection of subsets (or "hyperedges") of the vertex set V.

Definition 4.3 (General Separable Convex Program) *A general separable convex program is an optimization problem of the form*

$$\begin{array}{ll} \underset{x}{\text{minimize}} & F(x) \triangleq \sum_{i \in V} f_i(x_i) + \sum_{C \in \mathcal{C}} f_C(x_C) \\ \text{subject to} & x \in \mathbb{R}^V, \end{array} \tag{4.16}$$

where the factors $\{f_i(\cdot)\}$ are strictly convex, coercive, and twice continuously differentiable, the factors $\{f_C(\cdot)\}$ are convex and twice continuously differentiable, and

$$M \triangleq \min_i \inf_x \frac{\partial^2}{\partial x_i^2} F(x) > 0.$$

In this setting, as in Section 2.4.1, the min-sum algorithm operates by passing messages between vertices and hyperedges. In particular, denote the set of neighbor hyperedges to a vertex $i \in V$ by $\partial i \triangleq \{C \in \mathcal{C} : \ i \in C\}$. The min-sum update equations take the form

$$\begin{aligned} J_{i \to C}^{(t+1)}(x_i) &\triangleq f_i(x_i) + \sum_{C' \in \partial i \setminus C} J_{C' \to i}^{(t)}(x_i) + \kappa_{i \to C}^{(t)}, \\ J_{C \to i}^{(t+1)}(x_i) &\triangleq \min_{x_{C \setminus i}} \ f_C(x_C) + \sum_{i' \in C \setminus i} J_{i' \to C}^{(t+1)}(x_{i'}) + \kappa_{C \to i}^{(t)}. \end{aligned} \tag{4.17}$$

Local objective functions and estimates of the optimal solution are defined by

$$\begin{aligned} b_i^{(t)}(x_i) &\triangleq f_i(x_i) + \sum_{C \in \partial i} J_{C \to i}^{(t)}(x_i), \\ x_i^{(t)} &\triangleq \operatorname*{argmin}_{x_i} b_i^{(t)}(x_i). \end{aligned}$$

We will make the following assumption on the initial messages:

Assumption 4.2 (Min-Sum Initialization) *Assume that the each initial message $J_{C \to j}^{(0)}(\cdot)$ is twice continuously differentiable and that there exists some $z_{C \to j} \in \mathbb{R}^{C \setminus i}$ with*

$$\frac{d^2}{dx_j^2} J_{C \to j}^{(0)}(x_j) \geq \frac{\partial^2}{\partial x_j^2} f_C(x_j, z_{C \to j}), \quad \forall \ x_j \in \mathbb{R}.$$

Then, we have the following analog of Theorem 4.1:

Theorem 4.2 *Consider a general separable convex program. Assume that either:*

(i) The objective function $F(x)$ is (λ, w)-scaled diagonally dominant, and each pair of vertices $i, j \in V$ participate in at most one common factor. That is,

$$|\{C \in \mathcal{C} : \ (i,j) \subset C\}| \leq 1, \quad \forall \ i, j \in V.$$

(ii) The factors $\{f_C(\cdot)\}$ are individually (λ, w)-scaled diagonally dominant, in the sense that there exists a scalar $\lambda \in (0,1)$ and a vector $w \in \mathbb{R}^V$, with $w > 0$, so that for all $C \in \mathcal{C}$, $i \in C$, and $x_C \in \mathbb{R}^C$,

$$\sum_{j \in C \setminus i} w_j \left| \frac{\partial^2}{\partial x_i \partial x_j} f_C(x_C) \right| \leq \lambda w_i \frac{\partial^2}{\partial x_i^2} f_C(x_C).$$

Assume that the min-sum algorithm is initialized in accordance with Assumption 4.2. Define the constant

$$K \triangleq \frac{1}{M} \frac{\max_u w_u}{\min_u w_u}.$$

Then, the iterates of the min-sum algorithm satisfy

$$\|x^{(t)} - x^*\|_\infty \leq K \frac{\lambda^t}{1-\lambda} \sum_{C \in \mathcal{C}} \sum_{v \in C} \left| \frac{d}{dx_v} J^{(0)}_{C \to v}(x^*_v) - \frac{\partial}{\partial x_v} f_C(x^*_C) \right|.$$

Hence,

$$\lim_{t \to \infty} x^{(t)} = x^*.$$

Proof. This result can be proved using the same method as Theorem 4.1. The main modification required is the development of a suitable analog of Lemma 4.1. In the general case, scaled diagonal dominance of the computation tree does not follow from scaled diagonal dominance of the objective function $F(x)$. However, it is easy to verify that either of the hypotheses (i) or (ii) imply scaled diagonal dominance of the computation tree. The balance of the proof proceeds as in Section 4.1.4. ■

4.3 Asynchronous Convergence

The convergence results of Theorems 4.1 and 4.2 assumed a synchronous model of computation. That is, each message is updated at every time step in parallel. The min-sum update equations (4.3) and (4.17) are naturally decentralized. If we consider the application of the min-sum algorithm in distributed contexts, it is necessary to consider convergence under an *asynchronous* model of computation, as in Section 2.3.3. In this section, we will establish that Theorems 4.1 and 4.2 extend to an asynchronous setting, under the assumption of *total asynchronism* (Assumption 2.1).

Without loss of generality, consider the pairwise case of Theorem 4.1. This can be extended to the totally asynchronous setting as follows: the update equations take the form (2.15)–(2.19). We can repeat the construction of the computation tree in Section 4.1.3. As in the synchronous case, the initial messages only impact the leaves of computation tree. The total asynchronism assumption guarantees that these leaves are, eventually, arbitrarily far away from the root of the computation tree. The arguments in Lemma 4.3 then imply that the optimal value at the root of the computation tree is insensitive to the choice of initial messages. Convergence follows, as in Section 4.1.4.

The scaled diagonal dominance requirement of our convergence result is similar to conditions required for the totally asynchronous convergence of other optimization algorithms. Consider, for example, a decentralized coordinate descent algorithm. Here, the processor associated with vertex i maintains an estimate $x_i^{(t)}$ of the ith component of the optimal solution at time t. These estimates are updated at a sequence of times T^i, so that

$$x_i^{(t+1)} \triangleq \underset{x_i}{\operatorname{argmin}} \; f_i(x_i) + \sum_{u \in N(i)} f_{oi}(x_u^{(\tau_{u \to i}(t))}, x_i),$$

if $t \in T^i$, and $x_i^{(t+1)} \triangleq x_i^{(t)}$, otherwise. Here, $\tau_{u \to i}(t)$ is the time at which the most recent message received by vertex i from a neighboring vertex u was computed.

Similarly, consider a decentralized gradient method, where

$$x_i^{(t+1)} \triangleq x_i^{(t)} - \alpha \frac{\partial}{\partial x_i} \left(f_i\left(x_i^{(t)}\right) + \sum_{u \in N(i)} f_{ui}\left(x_u^{(\tau_{u \to i}(t))}, x_i^{(t)}\right) \right),$$

if $t \in T^i$, for some small positive step size α, and $x_i^{(t+1)} \triangleq x_i^{(t)}$, otherwise. These methods are not guaranteed to converge for arbitrary pairwise separable convex optimization problems. Typically, some sort of diagonal dominance condition is needed [14].

4.4 Implementation

The convergence theory we have presented elucidates properties of the min-sum algorithm and builds a bridge to the more established areas of convex analysis and optimization. However, except in very special cases, the algorithm as we have formulated it cannot be implemented on a digital computer because the messages that are computed and stored are functions over continuous domains. In this section, we present two variations that can be implemented to approximate behavior of the min-sum algorithm. For simplicity, we restrict attention to the case of the synchronous min-sum algorithm applied to pairwise separable convex programs.

Our first approach approximates messages using quadratic functions and can be viewed as a hybrid between the min-sum algorithm and Newton's method. It is easy to show that, if the single-variable factors $\{f_i(\cdot)\}$ are positive definite quadratics and the pairwise factors $\{f_{ij}(\cdot,\cdot)\}$ are positive semidefinite quadratics, then min-sum updates map quadratic messages to quadratic messages. The algorithm we propose here maintains a running estimate $\tilde{x}^{(t)}$ of the optimal solution, and at each time approximates each factor by a second-order Taylor expansion. In particular, let $\tilde{f}_i^{(t)}(\cdot)$ be the second-order Taylor expansion of $f_i(\cdot)$ around $\tilde{x}_i^{(t)}$ and let $\tilde{f}_{ij}^{(t)}(\cdot,\cdot)$ be the second-order Taylor expansion of $f_{ij}(\cdot,\cdot)$ around $(\tilde{x}_i^{(t)}, \tilde{x}_j^{(j)})$. Quadratic messages are updated according to

$$J_{i \to j}^{(t+1)}(x_j) \triangleq \min_{x_i} \ \kappa_{i \to j}^{(t+1)} + \tilde{f}_i^{(t)}(x_i) + \tilde{f}_{ij}^{(t)}(x_i, x_j) + \sum_{u \in N(i) \setminus j} J_{u \to i}^{(t)}(x_i), \tag{4.18}$$

where running estimates of the optimal solution are generated according to

$$\tilde{x}_i^{(t+1)} = \operatorname*{argmin}_{x_i} \ \tilde{f}_i^{(t+1)}(x_i) + \sum_{u \in N(i)} J_{u \to i}^{(t+1)}(x_i). \tag{4.19}$$

Note that the message update equation (4.18) takes the form of a Ricatti equation for a scalar system, which can be carried out efficiently. Further, each optimization problem (4.19) is a scalar unconstrained convex quadratic program.

A second approach makes use of a piecewise-linear approximation to each message. Let us assume knowledge that the optimal solution x^* is in a closed bounded set $[-B, B]^n$. Let $\mathcal{S} = \{\hat{x}_1, \ldots, \hat{x}_m\} \subset [-B, B]$, with $-B = \hat{x}_1 < \cdots < \hat{x}_m = B$, be a set of points where the linear pieces begin and end. Our approach applies the min-sum update equation to compute values at these points. Then, an approximation to the min-sum message is constructed via linear interpolation between consecutive points or extrapolation beyond the end points. In particular, the algorithm takes the form

$$J_{i \to j}^{(t+1)}(x_j) = \min_{x_i \in [-B,B]} \ \kappa_{i \to j}^{(t)} + f_i(x_i) + f_{ij}(x_i, x_j) + \sum_{u \in N(i) \setminus j} \hat{J}_{u \to i}^{(t)}(x_i), \tag{4.20}$$

for $x_j \in \mathcal{S}$, where

$$\hat{J}_{u \to i}^{(t)}(x_i) = \max_{1 \le k \le m-1} \ \frac{(\hat{x}_{k+1} - x_i) J_{u \to i}^{(t)}(\hat{x}_{k+1}) + (x_i - \hat{x}_k) J_{u \to i}^{(t)}(\hat{x}_k)}{\hat{x}_{k+1} - \hat{x}_k}, \tag{4.21}$$

for all $x_i \in \mathbb{R}$. As opposed to the case of quadratic approximations, where each message is parameterized by two numerical values, the number of parameters for each piecewise linear message grows with m. Hence, we anticipate that for fine-grain approximations, our second approach is likely to require greater computational resources. On the other hand, piecewise linear approximations may extend more effectively to nonconvex problems, since nonconvex messages are unlikely to be well-approximated by convex quadratic functions.

4.5 Open Issues

There are many open questions in the theory of message-passing algorithms. They fuel a growing research community that cuts across communications, artificial intelligence, statistical physics, theoretical computer science, and operations research. This chapter has focused on application of the min-sum message-passing algorithm to convex programs, and even in this context a number of interesting issues remain unresolved.

Our proof technique establishes convergence under total asynchronism assuming a scaled diagonal dominance condition. With such a flexible model of asynchronous computation, convergence results for local search algorithms such as gradient descent and coordinate descent also require similar diagonal dominance assumptions. On the other hand, for the *partially asynchronous* setting [14], where communication delays and times between successive updates are bounded, such assumptions are no longer required to guarantee convergence of local search algorithms. It would be interesting to see whether convergence of the min-sum algorithm under partial asynchronism can be established in the absence of scaled diagonal dominance. This is especially important since many practical convex optimization problems, such as the convex resource allocation problems described in Chapter 3, do not satisfy the scaled diagonal dominance condition.

Another direction will be to assess practical value of the min-sum algorithm for convex optimization problems. This calls for theoretical or empirical analysis of convergence and convergence times for implementable variants such as those proposed in the previous section. The convergence time results for the special case reported in Chapter 5 may provide a starting point. Our expectation is that for most relevant centralized optimization problems, the min-sum algorithm will be more efficient than gradient descent or coordinate descent but fall short of Newton's method. On the other hand, Newton's method does not decentralize gracefully, so in applications that call for decentralized solution, the min-sum algorithm may prove to be useful.

Finally, it would be interesting to explore whether ideas from this chapter can be helpful in analyzing behavior of the min-sum algorithm for nonconvex programs. It is encouraging that convex optimization theory has more broadly proved to be useful in designing and analyzing approximation methods for nonconvex programs.

5

CONSENSUS PROPAGATION

Consider a network described by a connected, undirected graph (V, E). Each of $n \triangleq |V|$ vertices observes a real number. Denote the number observed by a vertex $i \in V$ by $y_i \in \mathbb{R}$. Each vertex i contains a processor and has the ability to communicate only with neighboring vertices in the graph. The goal of the system is to compute the average

$$\bar{y} \triangleq \frac{1}{n} \sum_{i \in V} y_i.$$

One might imagine many schemes for computing the average $\bar{y}$. For example, all the vertices might communicate their observations to a single, central processor, and this processor can compute the average. Such a method requires significant coordination and overhead, however. A mechanism would be required for election of the central processor, and each of the rest of the processors would need the ability to transmit data along some path to this processor, for example, along a spanning tree or some other coordination structure. We are interested in a different class of methods, where the computation of $\bar{y}$ must be carried out in a distributed and asynchronous fashion. In such cases, it is required that any individual vertex in the network perform only simple, localized computations, and require knowledge of and communication with only its neighbors in the network. We refer to this as the *distributed consensus* problem.

The design of scalable distributed protocols for distributed consensus has received much recent attention and is motivated by a variety of potential needs.

In both wireless sensor and peer-to-peer networks, for example, there is interest in simple protocols for computing aggregate statistics (see, e.g., [37, 50, 51, 94, 8, 39, 64]), and averaging enables computation of several important ones. Further, averaging serves as a primitive in the design of more sophisticated distributed information processing algorithms. For example, a maximum likelihood estimate can be produced by an averaging protocol if each vertex's observations are linear in variables of interest and noise is Gaussian [93]. Averaging protocols are central to policy-gradient-based methods for distributed optimization of network performance [57]. Other applications include load balancing [26, 68, 28], clock synchronization [48, 92], and coordinated control of autonomous agents [38, 49, 69, 66, 72, 82].

In this chapter we propose and analyze a new protocol — *consensus propagation* — for distributed averaging. Consensus propagation is a messaging-passing algorithm. In particular, it is an application of the min-sum algorithm to the solution of a particular, unconstrained quadratic optimization problem. As such, the protocol can operate asynchronously and requires only simple iterative computations at individual vertices and communication of parsimonious messages between neighbors. There is no central hub that aggregates information. Each vertex only needs to be aware of its neighbors — no further information about the network topology is required. There is no need for construction of a specially-structured overlay network such as a spanning tree. It is worth discussing two previously proposed and well-studied protocols that also exhibit these features:

1. *(probabilistic counting)* This protocol is based on ideas for counting distinct elements of a database or stream [30, 7] and was adapted to produce a protocol for averaging [24]. The outcome is random, with variance that becomes arbitrarily small as the number of vertices grows. However, for moderate numbers of vertices, say tens of thousands, high variance makes the protocol impractical. The protocol can be repeated in parallel and results combined

in order to reduce variance, but this leads to onerous memory and communication requirements. Convergence time of the protocol is analyzed in [67].

2. *(linear consensus)* In linear consensus protocols, each vertex maintains a current estimate of the average, and each estimate is iteratively updated according to a linear combination the estimates of in a local neighborhood of the vertex. One example of such a protocol is the *pairwise averaging* protocol, where, at each time, a single pair of vertices communicate, they revise their estimates to both take on the mean of their previous estimates. Convergence of such linear consensus protocols in a very general model of asynchronous computation and communication was established in [87], and there has been significant follow-on work, a recent sample of which is [15]. Recent work on pairwise averaging [41, 20] has studied the convergence rate and its dependence on network topology and how pairs of vertices are sampled. Here, sampling is governed by a certain doubly stochastic matrix, and the convergence rate is characterized by its second-largest eigenvalue.

In terms of convergence rate, probabilistic counting dominates both linear consensus and consensus propagation in the asymptotic regime. However, consensus propagation and linear consensus are likely to be more effective in moderately-sized networks (up to hundreds of thousands or perhaps even millions of vertices). Further, these two protocols are both naturally studied as iterative matrix algorithms. As such, linear consensus will serve as a baseline to which we will compare consensus propagation.

With this background, let us discuss the primary contributions of this chapter:

1. *We propose consensus propagation, a new distributed and asynchronous protocol for averaging.*

2. *We prove that consensus propagation converges.*

3. *We characterize the convergence time in regular graphs of the synchronous version of consensus propagation in terms of the mixing time of a certain Markov chain over edges of the graph.*

4. *We explain why the convergence time of consensus propagation scales more gracefully with the number of vertices than does that of linear consensus, and for certain classes of graphs, we quantify the improvement.*

5.1 Problem Formulation

Consider a connected, undirected graph (V, E) with $n \triangleq |V|$ vertices. For each vertex $i \in V$, denote the set of vertices neighboring i by $N(i) \triangleq \{j \in V : (i,j) \in E\}$. Denote the set of edges with direction distinguished by $\vec{E} \triangleq \{(i,j) \in V \times V : i \in N(j)\}$.

Each vertex $i \in V$ is assigned a real number $y_i \in \mathbb{R}$. The goal of consensus propagation is for each vertex to obtain an estimate of the average $\bar{y} \triangleq \sum_{i \in V} y_i/n$, through an asynchronous distributed protocol in which each vertex carries out simple computations and communicates parsimonious messages to its neighbors.

Consensus propagation is parameterized by a scalar $\beta > 0$ and a symmetric, nonnegative matrix $Q \in \mathbb{R}_+^{V \times V}$ with $Q_{ij} > 0$ if and only if $i \neq j$ and $(i,j) \in E$. It is a special case of the min-sum algorithm, applied to the unconstrained quadratic optimization problem

$$\begin{array}{lll} \underset{x}{\text{minimize}} & F(x) \triangleq & \sum_{i \in V} (x_i - y_i)^2 + \sum_{(i,j) \in E} \beta Q_{ij} (x_i - x_j)^2 \\ \text{subject to} & & x \in \mathbb{R}^V. \end{array} \tag{5.1}$$

The objective function is a positive definite quadratic function, hence there exists a unique global minimizer $x^\beta \in \mathbb{R}^V$. Further, this problem is a pairwise separable convex program, as defined in Chapter 4.

The first term in the objective function forces each component x_i^β to be close to the corresponding value y_i. The second term forces each component x_i^β to be close

to the values of components adjacent in the graph — it is minimized when all the components of x are equal. For large values of β, the second term will dominate, and one might expect the optimal solution x^β to approach the optimal solution for the optimization problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \sum_{i \in V} (x_i - y_i)^2 \\ \text{subject to} & x_i = x_j, \qquad \forall\ (i,j) \in E. \end{array} \tag{5.2}$$

It is easy to see that (5.2) is minimized when $x_i = \bar{y}$, for all $i \in V$. The following theorem makes this argument precise.

Theorem 5.1 *For any value of $\beta > 0$,*

$$\frac{1}{n} \sum_i x_i^\beta = \bar{y}.$$

Further, for all $i \in V$,

$$\lim_{\beta \to \infty} x_i^\beta = \bar{y}.$$

Proof. Define the positive semidefinite matrix $\Gamma \in \mathbb{R}^{V \times V}$ so that

$$x^\top \Gamma x = \sum_{(i,j) \in E} Q_{ij} (x_i - x_j)^2 .$$

Then, the optimization program (5.1) can be written as

$$\underset{x \in \mathbb{R}^V}{\text{minimize}} \quad \|x - y\|^2 + \beta x^\top \Gamma x.$$

The first order conditions for optimality imply that $(I + \beta\Gamma)x^\beta = y$. Define $\mathbf{1} = (1, \ldots, 1)^\top \in \mathbb{R}^n$. Since the graph is connected, it follows from the definition of Γ that $\Gamma\mathbf{1} = 0$. Hence, $\mathbf{1}^\top x^\beta / n = \mathbf{1}^\top y / n = \bar{y}$.

Let U be an orthogonal matrix and D a diagonal matrix that form a spectral decomposition of Γ, that is $\Gamma = U^\top D U$. Then, we have $x^\beta = U^\top (I + \beta D)^{-1} U y$. It is clear that Γ has eigenvalue 0 with multiplicity 1 and corresponding (normalized)

eigenvector $\mathbf{1}/\sqrt{n}$, and that all other eigenvalues $d_2, \ldots, d_n$ of Γ are positive. Then, if $D = \operatorname{diag}(0, d_2, \ldots, d_n)$,

$$\begin{aligned}\lim_{\beta\to\infty} x^\beta &= \lim_{\beta\to\infty} U^\top \operatorname{diag}(1, 1/(1+\beta d_2), \ldots, 1/(1+\beta d_n))Uy \\ &= (\mathbf{1}/\sqrt{n})(\mathbf{1}/\sqrt{n})^\top y \\ &= \bar{y}\mathbf{1}.\end{aligned}$$

■

5.2 Message-Passing

Theorem 5.1 suggests that if β is sufficiently large, then each component x_i^β of the optimal solution to (5.1) can be used as an estimate of $\bar{y}$. Hence, one can apply the min-sum algorithm to attempt to solve the optimization problem (5.1), and use the resulting solution as an estimate of $\bar{y}$.

In particular, we can decompose the objective function $F(\cdot)$ as

$$F(x) = \sum_{i\in V} f_i(x_i) + \sum_{(i,j)\in E} f_{ij}(x_i, x_j),$$

where

$$\begin{aligned} f_i(x_i) &\triangleq (x_i - y_i)^2, \quad \forall\ i \in V, \\ f_{ij}(x_i, x_j) &\triangleq \beta Q_{ij}\,(x_i - x_j)^2, \quad \forall\ (i,j) \in E. \end{aligned}$$

The min-sum algorithm, as described in Sections 2.3 and 2.3.3, computes messages between vertices in an asynchronous and iterative fashion. For the moment, consider the synchronous case. The message from a vertex $i \in V$ to a neighboring vertex $j \in N(i)$ would be calculated according to (2.12). In this case, this becomes

$$\begin{aligned} J_{i\to j}^{(t+1)}(x_j) = \min_{x_i}\ & \kappa_{i\to j}^{(t)} + (x_i - y_i)^2 + \beta Q_{ij}(x_i - x_j)^2 \\ & + \sum_{u\in N(i)\setminus j} J_{u\to i}^{(t)}(x_i). \end{aligned} \tag{5.3}$$

Assume that each message is parameterized as a quadratic function, so that

$$J_{i\to j}^{(t)}(x_j) = K_{i\to j}^{(t)}\left(x_i - \mu_{i\to j}^{(t)}\right)^2, \tag{5.4}$$

given scalars $K_{i\to j}^{(t)} \in \mathbb{R}_+$ and $\mu_{i\to j}^{(t)} \in \mathbb{R}$, for each $i \in V$, $j \in N(i)$, and $t \geq 0$. Then, the update equation (5.3) also yields a quadratic function, and can directly be expressed in terms of the parameters as

$$K_{i\to j}^{(t+1)} = \frac{1 + \sum_{u\in N(i)\setminus j} K_{u\to i}^{(t)}}{1 + \frac{1}{\beta Q_{ij}}\left(1 + \sum_{u\in N(i)\setminus j} K_{u\to i}^{(t)}\right)}, \tag{5.5}$$

$$\mu_{i\to j}^{(t+1)} = \frac{y_i + \sum_{u\in N(i)\setminus j} K_{u\to i}^{(t)}\mu_{u\to i}^{(t)}}{1 + \sum_{u\in N(i)\setminus j} K_{u\to i}^{(t)}}. \tag{5.6}$$

Similarly, the min-sum algorithm computes an estimate of the optimal solution at vertex i according to (2.13)–(2.14). In this case, this becomes

$$x_i^{(t)} = \operatorname*{argmin}_{x_i}\ (x_i - y_i)^2 + \sum_{u\in N(i)} J_{u\to i}^{(t)}(x_i).$$

Applying the quadratic parametrization (5.4), it is easy to see that this is equivalent to

$$x_i^{(t)} = \frac{y_i + \sum_{u\in N(i)} K_{u\to i}^{(t)}\mu_{u\to i}^{(t)}}{1 + \sum_{u\in N(i)} K_{u\to i}^{(t)}}. \tag{5.7}$$

5.2.1 Intuitive Interpretation

The message-passing equations (5.5)–(5.7) have a natural interpretation. Consider the special case of a singly-connected graph (a connected graph where there are no cycles present). As illustrated in Figure 5.1, for any vertex $i \in V$ with a neighbor $j \in N(i)$, there is a set $S_{i\to j} \subset V$ of vertices, with $i \in S_{i\to j}$, that can transmit information to $S_{j\to i} \triangleq V \setminus S_{i\to j}$, with $j \in S_{j\to i}$, only through the link from vertex i to vertex j. This follows from the fact that the graph has no cycles. In order for vertices in $S_{j\to i}$ to compute $\overline{y}$, they must at least be provided with the average

$$\mu_{i\to j}^{*} \triangleq \frac{1}{|S_{i\to j}|}\sum_{u\in S_{i\to j}} y_u,$$

among observations at vertices in $S_{i\to j}$ and the cardinality $K^*_{i\to j} \triangleq |S_{i\to j}|$. Similarly, in order for vertices in $S_{i\to j}$ to compute $\overline{y}$, they must at least be provided with the average

$$\mu^*_{j\to i} \triangleq \frac{1}{|S_{j\to i}|} \sum_{u\in S_{j\to i}} y_u,$$

among observations at vertices in $S_{j\to i}$ and the cardinality $K^*_{j\to i} = |S_{j\to i}|$. These values must be communicated through the link between vertex j and vertex i.

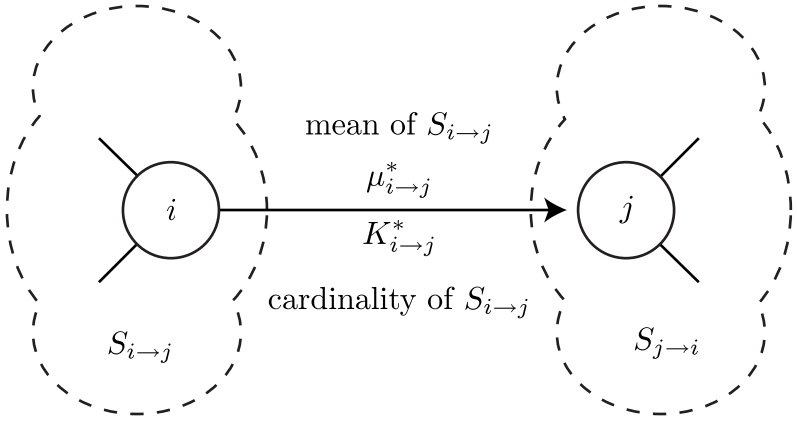


Figure 5.1 Interpretation of messages in a singly-connected graph with $\beta = \infty$.

Define messages $\mu^{(t)}_{i\to j} \in \mathbb{R}$ and $K^{(t)}_{i\to j} \in \mathbb{R}_+$ that are transmitted from vertex i to vertex j at time t. These messages are iterative estimates of the quantities $\mu^*_{i\to j}$ and $K^*_{i\to j}$. They evolve according to

$$\mu^{(t+1)}_{i\to j} \triangleq \frac{y_i + \sum_{u\in N(i)\setminus j} K^{(t)}_{u\to i}\mu^{(t)}_{u\to i}}{1 + \sum_{u\in N(i)\setminus j} K^{(t)}_{u\to i}}, \qquad \forall\ (i,j) \in \vec{E}, \tag{5.8a}$$

$$K^{(t+1)}_{i\to j} \triangleq 1 + \sum_{u\in N(i)\setminus j} K^{(t)}_{u\to i}, \qquad \forall\ (i,j) \in \vec{E}. \tag{5.8b}$$

At each time t, each vertex i computes an estimate of the global average $\bar{y}$ according to

$$x^{(t)}_i \triangleq \frac{y_i + \sum_{u\in N(i)} K^{(t)}_{u\to i}\mu^{(t)}_{u\to i}}{1 + \sum_{u\in N(i)} K^{(t)}_{u\to i}}.$$

Assume that the algorithm is initialized with $K^{(0)} \triangleq 0$. A simple inductive argument shows that at each time $t \geq 1$, $\mu_{i\to j}^{(t)}$ is the average among observations at the vertices in the set $S_{i\to j}$ that are at a distance less than or equal to t from vertex i. Furthermore, $K_{i\to j}^{(t)}$ is the cardinality of this collection of vertices. Since any vertex in $S_{i\to j}$ is at a distance from vertex i that it at most the diameter of the graph, if t is greater that the diameter of the graph, we have $K^{(t)} = K^*$ and $\mu^{(t)} = \mu^*$. Thus, for any $i \in V$, and t sufficiently large,

$$x_i^{(t)} = \frac{y_i + \sum_{u\in N(i)} K_{u\to i}^* \mu_{u\to i}^*}{1 + \sum_{u\in N(i)} K_{u\to i}^*} = \overline{y}.$$

So, $x_i^{(t)}$ converges to the global average $\overline{y}$. Further, this simple algorithm converges in as short a time as is possible, since the diameter of the graph is the minimum amount of time for the two most distance vertices to communicate.

Now, suppose that the graph has cycles. For any edge $(i, j) \in E$ that is part of a cycle, $K_{i\to j}^{(t)} \to \infty$. Hence, the algorithm does not converge. A heuristic fix might be to compose the iteration (5.8b) with one that attenuates:

$$\tilde{K}_{i\to j}^{(t)} \triangleq 1 + \sum_{u\in N(i)\setminus j} K_{u\to i}^{(t-1)},$$

$$K_{i\to j}^{(t)} \triangleq \frac{\tilde{K}_{i\to j}^{(t)}}{1 + \tilde{K}_{i\to j}^{(t)}/(\beta Q_{ij})}.$$

Here, $Q_{ij} > 0$ and $\beta > 0$ are positive constants. We can view the unattenuated algorithm as setting $\beta = \infty$. In the attenuated algorithm, the message is essentially unaffected when $\tilde{K}_{i\to j}^{(t)}/(\beta Q_{ij})$ is small but becomes increasingly attenuated as $\tilde{K}_{i\to j}^{(t)}$ grows. This is exactly the kind of attenuation carried out by the min-sum algorithm. Understanding why this kind of attenuation leads to desirable results is a subject of our analysis.

5.3 General Algorithm

Consensus propagation is the distributed and asynchronous application of the message-passing updates (5.5)–(5.7). In particular, every vertex $i \in V$ occasionally computes outgoing messages for each neighboring vertex $j \in N(i)$. These messages evolve over time, and each consist of two numerical parameters. Denote by $K_{i\to j}^{(t)} \in \mathbb{R}_+$ and $\mu_{i\to j}^{(t)} \in \mathbb{R}$ the values of the parameters for the message from vertex i to vertex j most recently computed at or before time t. Vertices will occasionally communicate with their neighbors and transmit the most recently computed message.

However, at each time t, each vertex i will have stored in its memory the most recent message parameters received from each neighbor:

$$\left\{\mu_{u\to i}^{(\tau_{u\to i}(t))},\ K_{u\to i}^{(\tau_{u\to i}(t))} :\ u \in N(i)\right\}. \tag{5.9}$$

Here, $0 \le \tau_{i\to j}(t) \le t$ is the time that the most recent message parameters, received at or before time t by vertex j from vertex i, was computed.

Each vertex i will compute new messages for a given neighboring vertex $j \in N(i)$ at a set of times $T^{i\to j}$. Thus, if $t \in T^{i\to j}$, the messages evolve according to

$$K_{i\to j}^{(t+1)} \triangleq \frac{1 + \sum_{u\in N(i)\setminus j} K_{u\to i}^{(\tau_{u\to i}(t))}}{1 + \frac{1}{\beta Q_{ij}}\left(1 + \sum_{u\in N(i)\setminus j} K_{u\to i}^{(\tau_{u\to i}(t))}\right)}, \tag{5.10}$$

$$\mu_{i\to j}^{(t+1)} \triangleq \frac{y_i + \sum_{u\in N(i)\setminus j} K_{u\to i}^{(\tau_{u\to i}(t))}\mu_{u\to i}^{(\tau_{u\to i}(t))}}{1 + \sum_{u\in N(i)\setminus j} K_{u\to i}^{(\tau_{u\to i}(t))}}. \tag{5.11}$$

At times $t \notin T^i$, the outgoing messages from vertex i do not change. Thus,

$$K_{i\to j}^{(t+1)} \triangleq K_{i\to j}^{(t)}, \tag{5.12}$$

$$\mu_{i\to j}^{(t+1)} \triangleq \mu_{i\to j}^{(t)}. \tag{5.13}$$

At a set of times T^i, the vertex i will compute a new estimate $x_i^{(t)}$ of the average $\bar{y}$. This estimate is computed as a function of the set of most recent messages (5.9)

received from all neighboring vertices. The estimate evolves according to

$$x_i^{(t)} \triangleq \frac{y_i + \sum_{u \in N(i)} K_{u \to i}^{(\tau_{u \to i}(t))} \mu_{u \to i}^{(\tau_{u \to i}(t))}}{1 + \sum_{u \in N(i)} K_{u \to i}^{(\tau_{u \to i}(t))}}, \tag{5.14}$$

if $t \in T^i$, and

$$x_i^{(t+1)} \triangleq x_i^{(t)}, \tag{5.15}$$

otherwise.

Denote by $\mu^{(t)}$ and $K^{(t)}$ the vector of all message parameters at time t,

$$K^{(t)} \triangleq \left(K_{i \to j}^{(t)} : \ (i,j) \in \vec{E} \right),$$
$$\mu^{(t)} \triangleq \left(\mu_{i \to j}^{(t)} : \ (i,j) \in \vec{E} \right),$$

and by $x^{(t)}$ the vector of all estimates of the average at time t,

$$x^{(t)} \triangleq \left(x_i^{(t)} : \ i \in V \right).$$

5.4 Convergence

Given that consensus propagation is a special case of the min-sum algorithm applied to an unconstrained convex optimization problem, one might expect to use the results of Chapter 4 to establish convergence. Indeed, a quick examination of the objective function $F(\cdot)$ of the optimization program (5.1) reveals that it is $(\lambda, \mathbf{1})$-scaled diagonally dominant, where

$$\lambda \triangleq \max_{i \in V} \ \frac{\sum_{j \in N(i)} \beta Q_{ij}}{1 + \sum_{j \in N(i)} \beta Q_{ij}} < 1.$$

Hence, Theorem 4.1 can be applied to yield convergence of the min-sum algorithm. Moreover, as discussed in Section 4.3, this convergence is asynchronous. Then, the following theorem is established:

Theorem 5.2 *Assume that the message updates satisfy the total asynchronism condition of Assumption 2.1, and that the messages are initialized in accordance with Assumption 4.1. Then,*

$$\lim_{t\to\infty} x^{(t)} = x^{\beta}.$$

Theorem 5.2 guarantees convergence of the estimates generated by the consensus propagation algorithm to the optimal solution of (5.1). It is possible to, however, make a stronger statement. The following theorem guarantees that the messages themselves also converge to a unique fixed point of the iterations (5.5)–(5.6), and allows for arbitrary initial conditions for the convergence of the messages and the estimates.

Theorem 5.3 *The following hold:*

(i) There exist unique vectors $K^{\beta} \in \mathbb{R}_{+}^{\vec{E}}$ and $\mu^{\beta} \in \mathbb{R}^{\vec{E}}$ that satisfy

$$K_{i\to j}^{\beta} = \frac{1 + \sum_{u\in N(i)\setminus j} K_{u\to i}^{\beta}}{1 + \frac{1}{\beta Q_{ij}}\left(1 + \sum_{u\in N(i)\setminus j} K_{u\to i}^{\beta}\right)},$$
$$\mu_{i\to j}^{\beta} = \frac{y_i + \sum_{u\in N(i)\setminus j} K_{u\to i}^{\beta}\mu_{u\to i}^{\beta}}{1 + \sum_{u\in N(i)\setminus j} K_{u\to i}^{\beta}},$$

for all $i \in V$ and $j \in N(i)$.

(ii) Suppose that the message updates satisfy the total asynchronism condition of Assumption 2.1. Then, independent of the initial condition $(\mu^{(0)}, K^{(0)}) \in \mathbb{R}^{\vec{E}} \times \mathbb{R}_{+}^{\vec{E}}$,

$$\lim_{t\to\infty} K^{(t)} = K^{\beta}, \quad \text{and} \quad \lim_{t\to\infty} \mu^{(t)} = \mu^{\beta}.$$

(iii) For every $i \in V$,

$$x_i^{\beta} = \frac{y_i + \sum_{u\in N(i)} K_{u\to i}^{\beta}\mu_{u\to i}^{\beta}}{1 + \sum_{u\in N(i)} K_{u\to i}^{\beta}}.$$

Hence, under the assumptions of Part (ii),

$$\lim_{t\to\infty} x^{(t)} = x^{\beta}.$$

The proof of this theorem is deferred until Section 5.7.1, but it rests upon two ideas. First, notice that, according to the update equations (5.5)–(5.6), $K^{(t)}$ evolves independently of $\mu^{(t)}$. Hence, we analyze $K^{(t)}$ first. Following the work in [79], we prove that the updates (5.5) are monotonic. This property is used to establish convergence to a unique fixed point. Next, we analyze $\mu^{(t)}$ assuming that $K^{(t)}$ has already converged. Given fixed $K^{(t)}$, the update equations (5.6) for $\mu^{(t)}$ are linear, and we establish that they induce a contraction with respect to the maximum norm. This allows us to establish existence of a fixed point and both synchronous and asynchronous convergence.

5.5 Convergence Time for Regular Graphs

In this section, we will study the convergence time of synchronous consensus propagation, defined by (5.5)–(5.7).

For $\epsilon > 0$, we will say that an estimate $\tilde{x}$ of $\bar{y}$ is ϵ-accurate if

$$\|\tilde{x} - \bar{y}\mathbf{1}\|_{2,n} \leq \epsilon. \tag{5.16}$$

Here, for integer m, we set $\|\cdot\|_{2,m}$ to be the norm on $\mathbb{R}^m$ defined by $\|x\|_{2,m} \triangleq \|x\|_2/\sqrt{m}$. We are interested in the number of iterations required to obtain an ϵ-accurate estimate of the mean $\bar{y}$.

Note that we are primarily interested in how the performance of consensus propagation behaves over a series of problem instances as we scale the size of the graph. Since our measure of error (5.16) is absolute, we require that the set of values $\{y_i\}$ lie in some bounded set. Without loss of generality, we will take $y_i \in [0, 1]$, for all $i \in V$.

5.5.1 The Case of Regular Graphs

We will restrict our analysis of convergence time to cases where (V, E) is a d-regular graph, for $d \geq 2$. Extension of our analysis to broader classes of graphs remains

an open issue. We will also make the simplifying assumptions that

$$Q_{ij} \triangleq 1, \quad \forall\ (i,j) \in E, \tag{5.17}$$

$$\mu_{i\to j}^{(0)} \triangleq y_i, \quad \forall\ i \in V,\ j \in N(i), \tag{5.18}$$

$$K_{i\to j}^{(0)} \triangleq k_0, \quad \forall\ i \in V,\ j \in N(i), \tag{5.19}$$

for some scalar $k_0 \geq 0$.

In this restricted setting, the subspace of constant K vectors is invariant under the update (5.5). This implies that there is some scalar $k^\beta > 0$ so that $K_{i\to j}^{\beta} = k^\beta$, for all $i \in V$ and $j \in N(i)$. The value k^β is the unique solution to the fixed point equation

$$k^\beta = \frac{1 + (d-1)k^\beta}{1 + (1 + (d-1)k^\beta)/\beta}. \tag{5.20}$$

Given a uniform initial condition (5.19), we can study the sequence of iterates $\{K^{(t)}\}$ by examining the scalar sequence $\{k_t\}$, defined by

$$k_t \triangleq \frac{1 + (d-1)k_{t-1}}{1 + (1 + (d-1)k_{t-1})/\beta}. \tag{5.21}$$

In particular, we have $K_{i\to j}^{(t)} = k_t$, for all $i \in V$, $j \in N(i)$, and $t \geq 0$.

Similarly, in this setting, the equations for the evolution of $\mu^{(t)}$ take the special form

$$\begin{aligned}\mu_{i\to j}^{(t)} = &\frac{y_i}{1 + (d-1)k_{t-1}} \\ &+ \left(1 - \frac{1}{1 + (d-1)k_{t-1}}\right) \sum_{u \in N(i)\setminus j} \frac{\mu_{u\to i}^{(t-1)}}{d-1}.\end{aligned}$$

Defining $\gamma_t = 1/(1 + (d-1)k_t)$, we have, in vector form,

$$\mu^{(t)} = \gamma_{t-1}\hat{y} + (1 - \gamma_{t-1})\hat{P}\mu^{(t-1)}, \tag{5.22}$$

where $\hat{y} \in \mathbb{R}^{nd}$ is a vector with $\hat{y}_{i\to j} = y_i$ and $\hat{P} \in \mathbb{R}_+^{nd\times nd}$ is a doubly stochastic matrix. The matrix $\hat{P}$ corresponds to a Markov chain on the set of directed edges

$\vec{E}$. In this chain, a directed edge (i, j) transitions to a directed edge (u, i) with $u \in N(i) \setminus j$, with equal probability assigned to each such edge. As in (5.14), we associate each $\mu^{(t)}$ with an estimate $x^{(t)}$ of x^β according to

$$x^{(t)} = \frac{1}{1 + dk_t} y + \frac{dk_t}{1 + dk_t} A\mu^{(t)},$$

where $A \in \mathbb{R}_+^{n \times nd}$ is a matrix defined by $(A\mu)_j \triangleq \sum_{i \in N(j)} \mu_{i \to j}/d$.

5.5.2 The Cesàro Mixing Time

The update equation (5.22) suggests that the convergence of $\mu^{(t)}$ is intimately tied to a notion of mixing time associated with $\hat{P}$. Let $\hat{P}^\star$ be the Cesàro limit

$$\hat{P}^\star \triangleq \lim_{t \to \infty} \sum_{\tau=0}^{t-1} \hat{P}^\tau / t.$$

Define the *Cesàro mixing time* $\tau^\star$ by

$$\tau^\star \triangleq \sup_{t \geq 0} \left\| \sum_{\tau=0}^{t} (\hat{P}^\tau - \hat{P}^\star) \right\|_{2,nd}.$$

Here, $\|\cdot\|_{2,nd}$ is the matrix norm induced by the corresponding vector norm $\|\cdot\|_{2,nd}$. Since $\hat{P}$ is a stochastic matrix, $\hat{P}^\star$ is well-defined and $\tau^\star < \infty$. Note that, in the case where $\hat{P}$ is aperiodic, irreducible, and symmetric, $\tau^\star$ corresponds to the traditional definition of mixing time: the inverse of the spectral gap of $\hat{P}$.

5.5.3 Bounds on the Convergence Time

Let $\gamma^\beta = \lim_{t \to \infty} \gamma_t = 1/(1 + (d-1)k^\beta)$. With an initial condition $k_0 = k^\beta$, the update equation for $\mu^{(t)}$ becomes

$$\mu^{(t)} = \gamma^\beta \hat{y} + (1 - \gamma^\beta) \hat{P} \mu^{(t-1)}.$$

Since $\gamma^\beta \in (0, 1)$, this iteration is a contraction mapping, with contraction factor $1 - \gamma^\beta$. It is easy to show that γ^β is monotonically decreasing in β, and as such,

large values of β are likely to result in slower convergence. On the other hand, Theorem 5.1 suggests that large values of β are required to obtain accurate estimates of $\bar{y}$. To balance these conflicting issues, β must be appropriately chosen.

A time t^* is said to be an *ϵ-convergence time* if each estimate $x^{(t)}$ is ϵ-accurate for all $t \geq t^*$. The following theorem, whose proof is provided in Section 5.7.2, establishes a bound on the ϵ-convergence time of synchronous consensus propagation given an appropriately chosen β, as a function of ϵ and $\tau^\star$.

Theorem 5.4 *Suppose $k_0 \leq k^\beta$. If $d = 2$ there exists a $\beta = \Theta((\tau^\star/\epsilon)^2)$ and if $d > 2$ there exists a $\beta = \Theta(\tau^\star/\epsilon)$ such that some $t^* = O((\tau^\star/\epsilon)\log(\tau^\star/\epsilon))$ is an ϵ-convergence time.*

In the above theorem, k_0 is initialized arbitrarily so long as $k_0 \leq k^\beta$. Typically, one might set $k_0 = 0$ to guarantee this. Another case of particular interest is when $k_0 = k^\beta$, so that $k_t = k^\beta$ for all $t \geq 0$. In this case, the following theorem, whose proof is provided in Section 5.7.2, offers a better convergence time bound than Theorem 5.4.

Theorem 5.5 *Suppose $k_0 = k^\beta$. If $d = 2$ there exists a $\beta = \Theta((\tau^\star/\epsilon)^2)$ and if $d > 2$ there exists a $\beta = \Theta(\tau^\star/\epsilon)$ such that some $t^* = O((\tau^\star/\epsilon)\log(1/\epsilon))$ is an ϵ-convergence time.*

Theorems 5.4 and 5.5 suggest that initializing with $k_0 = k^\beta$ leads to an improvement in convergence time. However, in our computational experience, we have found that an initial condition of $k_0 = 0$ consistently results in *faster* convergence than $k_0 = k^\beta$. Hence, we suspect that a convergence time bound of $O((\tau^\star/\epsilon)\log(1/\epsilon))$ also holds for the case of $k_0 = 0$. Proving this remains an open issue.

5.5.4 Adaptive Mixing Time Search

The choice of β is critical in that it determines both the convergence time and the ultimate accuracy of the consensus propagation algorithm. This raises the question of how to choose β for a particular graph. The choices posited in Theorems 5.4

Algorithm 5.1 Synchronous consensus propagation on a d-regular graph with an adaptive mixing time search.

1: $k_0 \leftarrow 0$, $\mu^{(0)} \leftarrow \hat{y}$, $t \leftarrow 0$
2: **for** $\ell = 0$ to ∞ **do**
3: $\quad \tilde{\tau} \leftarrow 2^\ell$
4: $\quad$ Set β and t^* as indicated by Theorem 5.4, assuming $\tau^\star = \tilde{\tau}$
5: $\quad$ **for** $s = 1$ to t^* **do**
6: $\quad\quad$ Update messages according to

$$k_t \leftarrow \frac{1 + (d-1)k_{t-1}}{1 + (1 + (d-1)k_{t-1})/\beta},$$
$$\mu^{(t)} \leftarrow \frac{1}{1 + (d-1)k_{t-1}}\hat{y} + \frac{(d-1)k_{t-1}}{1 + (d-1)k_{t-1}}\hat{P}\mu^{(t-1)}.$$

7: $\quad\quad t \leftarrow t + 1$
8: $\quad$ **end for**
9: **end for**

and 5.5 require knowledge of $\tau^\star$, which may be both difficult to compute and also requires knowledge of the graph topology. This is at odds with our goal of developing a distributed protocol.

In order to address this concern, consider Algorithm 5.1, which is designed for the case of $d > 2$. It uses a doubling sequence of guesses $\tilde{\tau}$ for the Cesáro mixing time $\tau^\star$. Each guess leads to a choice of β and a number of iterations t^*. Note that the algorithm takes $\epsilon > 0$ as input.

Consider applying this procedure to a d-regular graph with fixed $d > 2$ but topology otherwise unspecified. Analysis that follows from Theorem 5.4 reveals that this procedure has an ϵ-convergence time of $O((\tau^\star/\epsilon)\log(\tau^\star/\epsilon))$. An entirely analogous algorithm can be designed for the case of $d = 2$.

We expect that many variations of this procedure can be made effective. Asynchronous versions would involve each vertex adapting a local estimate of the mixing time.

5.6 Comparison with Linear Consensus

In order to make the comparison with consensus propagation directly, we define a synchronous linear consensus algorithm as follows. Each vertex $i \in V$ maintains an estimate $x_i^{(t)} \in \mathbb{R}$ of the global average $\bar{y}$ that evolves over discrete time steps $t \geq 0$. These estimates are initialized so that $x_i^{(0)} = y_i$, for each vertex i. Given a doubly stochastic symmetric matrix $P \in \mathbb{R}^{n \times n}$, such that $P_{ij} = 0$ if $i \neq j$ and $(i, j) \notin E$, estimates evolve according to

$$x^{(t+1)} \triangleq P x^{(t)}. \tag{5.23}$$

Here, at each time t, a vertex i is computing a new estimate $x_i^{(t+1)}$ which is an average of the estimates at vertex i and its neighbors during the previous time-step. If the matrix P is aperiodic and irreducible, then

$$\lim_{t \to \infty} x^{(t)} = \lim_{t \to \infty} P^t y = \bar{y}\mathbf{1}.$$

Linear consensus can be viewed as a local search method for optimization. In particular, consider the optimization problem

$$\begin{array}{lll} \underset{x}{\text{minimize}} & G(x) \triangleq & \frac{1}{2} x^\top (I - P) x \\ \text{subject to} & & \mathbf{1}^\top x / n = \bar{y}, \\ & & x \in \mathbb{R}^V. \end{array} \tag{5.24}$$

By our assumptions on P (doubly stochastic, aperiodic, irreducible), this quadratic optimization program is uniquely minimized when $x = \bar{y}\mathbf{1}$. Assume that, at time t, we are given a vector $x^{(t)} \in \mathbb{R}^V$ that is feasible, i.e., $\mathbf{1}^\top x^{(t)} / n = \bar{y}$. If we apply a gradient descent update,

$$\begin{aligned} x^{(t+1)} &\triangleq x^{(t)} - \nabla G(x^{(t)}) \\ &= x^{(t)} - (I - P) x^{(t)} = P x^{(t)}. \end{aligned}$$

This is identical to the linear consensus update (5.23). The vector $x^{(t+1)}$ is also feasible for (5.24), since

$$\mathbf{1}^\top x^{(t+1)} / n = \mathbf{1}^\top P x^{(t)} / n = \mathbf{1}^\top x^{(t)} / n = \bar{y}.$$

Hence, we can interpret linear consensus as gradient descent in the feasible region of the optimization program (5.24).

5.6.1 Rate of Convergence

In the case of a singly-connected graph, synchronous consensus propagation converges exactly in a number of iterations equal to the diameter of the graph. Moreover, when $\beta = \infty$, this convergence is to the exact mean, as discussed in Section 5.2.1. This is the best one can hope for under any algorithm, since the diameter is the minimum amount of time required for a message to travel between the two most distant vertices. On the other hand, for a fixed accuracy ϵ, the worst-case number of iterations required by linear consensus on a singly-connected graph scales at least quadratically in the diameter [19].

The rate of convergence of linear consensus is governed by the relation $\|x^{(t)} - \bar{y}\mathbf{1}\|_{2,n} \leq \lambda_2^t$, where λ_2 is the second largest eigenvalue[1] of P. Let $\tau_2 \triangleq 1/\log(1/\lambda_2)$, and call it the *mixing time* of P. In order to guarantee ϵ-accuracy (independent of y), $t > \tau_2 \log(1/\epsilon)$ suffices and $t = \Omega(\tau_2 \log(1/\epsilon))$ is required.

Consider d-regular graphs and fix a desired error tolerance ϵ. The number of iterations required by consensus propagation is $\Theta(\tau^\star \log \tau^\star)$, whereas that required by linear consensus is $\Theta(\tau_2)$. Both mixing times depend on the size and topology of the graph. τ_2 is the mixing time of a process on vertices that transitions along edges whereas $\tau^\star$ is the mixing time of a process on directed edges that transitions towards vertices. An important distinction is that the former process is allowed to "backtrack" where as the latter is not. By this we mean that a sequence of states (i, j, i) can be observed in the vertex process, but the sequence $((i, j), (j, i))$ cannot be observed in the edge process. As we will now illustrate through an example, it is this difference that makes τ_2 larger than $\tau^\star$ and, therefore, linear consensus less efficient than consensus propagation.

[1]Here, we take the standard approach of ignoring the smallest, possibly negative eigenvalue of P. We will assume that this eigenvalue is smaller than λ_2 in magnitude. Note that a constant probability can be added to each self-loop of any particular matrix P so that this is true.

In the case of a cycle ($d = 2$) with an even number of vertices n, minimizing the mixing time over P results in $\tau_2 = \Theta(n^2)$ [18, 20]. For comparison, as demonstrated in the following theorem (whose proof is provided in Section 5.7.3), $\tau^\star$ is linear in n.

Theorem 5.6 *For the cycle with n vertices, $\tau^\star \leq n/\sqrt{2}$.*

Intuitively, the improvement in mixing time arises from the fact that the edge process moves around the cycle in a single direction and therefore travels distance t in order t iterations. The vertex process, on the other hand, is "diffusive" in nature. It randomly transitions back and forth among adjacent vertices, and requires order t^2 iterations to travel distance t. Nondiffusive methods have previously been suggested in the design of efficient algorithms for Markov chain sampling (see [27] and references therein).

The cycle example demonstrates a $\Theta(n/\log n)$ advantage offered by consensus propagation. Comparisons of mixing times associated with other graph topologies remains an issue for future analysis. Let us close by speculating on a uniform grid of n vertices over the m-dimensional unit torus. Here, $n^{1/m}$ is an integer, and each vertex has $2m$ neighbors, each a distance $n^{-1/m}$ away. With P optimized, it can be shown that $\tau_2 = \Theta(n^{2/m})$ [77]. We put forth a conjecture on $\tau^\star$.

Conjecture 5.1 *For the m-dimensional torus with n vertices,*

$$\tau^\star = \Theta(n^{(2m-1)/m^2}).$$

5.7 Proofs

In the section, we provide proofs for the main results of the chapter.

5.7.1 Proof of Theorem 5.3

For the proof of Theorem 5.3, it is helpful to define, for each $i \in j$ and $j \in N(i)$, functions corresponding to the updates in (5.5)–(5.6),

$$\mathcal{F}_{i\to j}(K) \triangleq \frac{1 + \sum_{u\in N(i)\backslash j} K_{u\to i}}{1 + \frac{1}{\beta Q_{ij}}\left(1 + \sum_{u\in N(i)\backslash j} K_{u\to i}\right)},$$
$$\mathcal{G}_{i\to j}(\mu, K) \triangleq \frac{y_i + \sum_{u\in N(i)\backslash j} K_{u\to i}\mu^{(t)}_{u\to i}}{1 + \sum_{u\in N(i)\backslash j} K_{u\to i}}.$$

We define the operators $\mathcal{F} : \mathbb{R}_+^{\vec{E}} \to \mathbb{R}_+^{\vec{E}}$ and $\mathcal{G} : \mathbb{R}^{\vec{E}} \times \mathbb{R}_+^{\vec{E}} \to \mathbb{R}^{\vec{E}}$ to correspond to the one-step synchronous updates of the K and μ parameter vectors, respectively,

$$\mathcal{F}(K)_{i\to j} \triangleq \mathcal{F}_{i\to j}(K), \quad \forall\ (i,j) \in \vec{E},$$
$$\mathcal{G}(\mu, K)_{i\to j} \triangleq \mathcal{G}_{i\to j}(\mu, K), \quad \forall\ (i,j) \in \vec{E}.$$

In order to establish Theorem 5.3, we will first study convergence of the parameters $K^{(t)}$, and subsequently the parameters $\mu^{(t)}$.

Convergence of $K^{(t)}$

Our analysis of the convergence of the inverse variance parameters follows the work of Van Roy and Rusmevichientong [79]. We begin with a fundamental lemma.

Lemma 5.1 *For each $(i,j) \in \vec{E}$, the following facts hold:*

(i) The function $\mathcal{F}_{ij}(\cdot)$ is continuous.

(ii) The function $\mathcal{F}_{ij}(\cdot)$ is monotonic. That is, if $K \leq K'$, where the inequality is interpreted component-wise, then $\mathcal{F}_{ij}(K) \leq \mathcal{F}_{ij}(K')$.

(iii) If $K'_{ij} = \mathcal{F}_{ij}(K)$, then $0 < K'_{ij} < \beta Q_{ij}$.

(iv) If $\alpha > 1$, then $\alpha\mathcal{F}_{ij}(K) > \mathcal{F}_{ij}(\alpha K)$.

Proof. Define the function $f : \mathbb{R}_+ \to \mathbb{R}_+$ by

$$f(x) = \frac{1}{\gamma + \dfrac{1}{1+x}},$$

where $\gamma > 0$. (i) follows from the fact that f is continuous. (ii) follows from the fact that $f(x)$ is strictly increasing. (iii) follows from the fact that $f(x) \in (0, 1/\gamma)$ for all $x \geq 0$. (iv) follows from the fact that $\alpha f(x) \geq f(\alpha x)$. ■

Lemma 5.2 *Let $H^{(0)} \in \mathbb{R}_+^{\vec{E}}$ be such that $\mathcal{F}_{ij}(H^{(0)}) \geq H^{(0)}$ for all $(i,j) \in \vec{E}$. Define $H^{(t)}$, for $t \geq 1$, by*

$$H^{(t)} \triangleq \mathcal{F}(H^{(t-1)}). \tag{5.25}$$

Then $H^{(t)}$ converges to a vector K^β such that $K^\beta = \mathcal{F}(K^\beta)$.

Similarly, assume that $H^{(0)}$ satisfies $\mathcal{F}_{ij}(H^{(0)}) \leq H^{(0)}$. Then $H^{(t)}$ converges to a vector K^β such that $K^\beta = \mathcal{F}(K^\beta)$.

Proof. Convergence follows from the fact that the iterates are component-wise bounded and monotonic. The limit point must be a fixed point by continuity. ■

Given the above lemma, we can establish existence of a unique fixed point.

Lemma 5.3 *The $\mathcal{F}$ operator has a unique fixed point K^β.*

Proof. Denote K^β to be the fixed point obtained by iterating (5.25) with initial condition $H^{(0)} = 0$, and let K' be some other fixed point. It is clear that $K^{(0)} < K'$, thus, by monotonicity, we must have $K^\beta \leq K'$. Define

$$\gamma = \inf\left\{\alpha \in [1, \infty) : \ K' \leq \alpha K^\beta\right\}.$$

It is clear that γ is well-defined since $0 < \{K^\beta_{ij}, K'_{ij}\} < \beta Q_{ij}$. Also, we must have $\gamma > 1$, since $K^\beta \neq K'$. Then,

$$K'_{ij} = \mathcal{F}_{ij}(K') \leq \mathcal{F}_{ij}(\gamma K^\beta) < \gamma \mathcal{F}_{ij}(K^\beta) = \gamma K^\beta_{ij}.$$

This contradicts the definition of γ. Hence, there is a unique fixed point. ■

Now, we can establish asynchronous convergence.

Lemma 5.4 *Assume that the message updates satisfy the total asynchronism condition of Assumption 2.1. Then, given an arbitrary initial condition $K^{(0)} \in \mathbb{R}_+^{\vec{E}}$, the parameters $K^{(t)}$ converge asynchronously, that is*

$$\lim_{t\to\infty} K^{(t)} = K^\beta.$$

Proof. First, define the vectors $\underline{K}^{(t)}$, for $t \geq 1$, by

$$\begin{aligned} \underline{K}^{(1)}_{i\to j} &\triangleq 0, \quad \forall\ (i,j) \in \vec{E}, \\ \underline{K}^{(t)} &\triangleq \mathcal{F}(\underline{K}^{(t-1)}), \quad \forall\ t \geq 2. \end{aligned}$$

Note that $\underline{K}^{(1)} \leq \mathcal{F}(\underline{K}^{(1)})$, thus, the sequence $\{\underline{K}^{(t)}\}$ is component-wise monotonically increasing and converges to K^β.

Similarly, define the vectors $\overline{K}^{(t)}$, for $t \geq 1$, by

$$\begin{aligned} \overline{K}^{(1)}_{i\to j} &\triangleq \beta Q_{ij}, \quad \forall\ (i,j) \in \vec{E}, \\ \overline{K}^{(t)} &\triangleq \mathcal{F}(\overline{K}^{(t-1)}), \quad \forall\ t \geq 2. \end{aligned}$$

Note that $\overline{K}^{(1)} \geq \mathcal{F}(\overline{K}^{(1)})$, thus, the sequence $\{\overline{K}^{(t)}\}$ is component-wise monotonically decreasing and converges to K^β.

Define, the set $\mathcal{K}^{(0)} = \mathbb{R}_+^{\vec{E}}$, and, for $t \geq 1$, the set

$$\mathcal{K}^{(t)} = \left\{ K \in \mathbb{R}_+^{\vec{E}} :\ \underline{K}^{(t)} \leq K \leq \overline{K}^{(t)} \right\},$$

Note that, for all $t \geq 0$, $\mathcal{K}^{(t+1)} \subset \mathcal{K}^{(t)}$. Furthermore,

$$\bigcap_{t=0}^{\infty} \mathcal{K}^{(t)} = \left\{ K^\beta \right\}.$$

Finally, if, by monotonicity, $K \in \mathcal{K}^{(t)}$, then $\mathcal{F}(K) \in \mathcal{K}^{(t+1)}$.

Since the asynchronous iterates $K^{(t)}$ are generated by asynchronous component-wise updates of the operator $\mathcal{F}(\cdot)$, and since the total asynchronism assumption holds, convergence of $K^{(t)}$ to K^β follows from Proposition 6.2.1 in [14]. ■

Properties of the Map $\mathcal{G}$

In this section, we will consider certain properties of the map $\mathcal{G}$.

Lemma 5.5 *There exists $\alpha \in (0,1)$ and $\epsilon_1 > 0$ so that*

(i) For all $\mu, \mu' \in \mathbb{R}^{\vec{E}}$,

$$\|\mathcal{G}(\mu, K^\beta) - \mathcal{G}(\mu', K^\beta)\|_\infty \leq \alpha \|\mu - \mu'\|_\infty.$$

(ii) If

$$\|K - K^\beta\|_\infty < \epsilon_1,$$

then, for all $\mu, \mu' \in \mathbb{R}^{\vec{E}}$,

$$\|\mathcal{G}(\mu, K) - \mathcal{G}(\mu', K)\|_\infty \leq \alpha \|\mu - \mu'\|_\infty.$$

Proof. Define

$$\bar{\alpha}(K) = \max_{\substack{(i,j)\in\vec{E} \\ u\in N(i)\setminus j}} \frac{K_{ui}}{1 + \sum_{u'\in N(i)\setminus j} K_{u'i}}.$$

Observing that $\bar{\alpha}(K^\beta) < 1$, Part (i) follows. By the continuity of $\bar{\alpha}(\cdot)$ in the neighborhood of K^β, Part (ii) follows. ■

Lemma 5.5 states that $\mathcal{G}(\cdot, K^\beta)$ is a maximum norm contraction. This leads to the following lemma.

Lemma 5.6 *The following hold:*

(i) There is unique fixed point μ^β such that

$$\mu^\beta = \mathcal{G}(\mu^\beta, K^\beta).$$

(ii) If $\|K - K^\beta\| < \epsilon_1$, then the operator $\mathcal{G}(\cdot, K)$ has a unique fixed point $\nu(K)$. That is,

$$\nu(K) = \mathcal{G}(\nu(K), K).$$

(iii) For any $\delta > 0$, there exists $\epsilon_2 \leq \epsilon_1$ so that if $\|K - K^\beta\|_\infty < \epsilon_2$, then

$$\|\nu(K) - \mu^\beta\|_\infty < \delta.$$

Proof. For Part (i), since $\mathcal{G}(\cdot, K^\beta)$ is a maximum norm contraction, existence of a unique fixed point μ^β follows from, for example, Proposition 3.1.1 in [14]. Part (ii) is established similarly.

For Part (iii), note for vectors K sufficiently close to K^β, the linear system of equations

$$\nu = \mathcal{G}(\nu, K)$$

over $\nu \in \mathbb{R}^{\vec{E}}$ is nonsingular, by Part (ii). Since the coefficients of this system of equations continuously converge as $K \to K^\beta$ to those of

$$\nu = \mathcal{G}(\nu, K^\beta),$$

we must have $\nu \to \mu^\beta$. ■

Convergence of $\mu^{(t)}$

In order to establish the asynchronous convergence of $\mu^{(t)}$, we need some additional notation.

For each time $t \geq 0$, define $E(t)$ to be the earliest time that messages computed prior to time t have been flushed from the system, that is

$$E(t) \triangleq \min \{s \geq 0 : \ \forall \ (i,j) \in \vec{E}, \ r \in T^{i \to j} \text{ with } r \geq s, \ \tau_{i \to j}(r) \geq t\}.$$

$E(t)$ exists by our total asynchronism assumption.

For all $(i,j) \in \vec{E}$ and $t \in T^{i \to j}$, define the vector $K^{(t), i \to j} \in \mathbb{R}_+^{\vec{E}}$ by

$$K_{u \to v}^{(t), i \to j} \triangleq \begin{cases} K_{u \to i}^{(\tau_{u \to i}(t))} & \text{if } u = i \text{ and } v \in N(i) \setminus j, \\ K_{u \to v}^{\beta} & \text{otherwise.} \end{cases}$$

and the vector $\mu^{(t),i\to j} \in \mathbb{R}^{\vec{E}}$ by

$$\mu_{u\to v}^{(t),i\to j} \triangleq \begin{cases} \mu_{u\to i}^{(\tau_{u\to i}(t))} & \text{if } u = i \text{ and } v \in N(i) \setminus j, \\ \mu_{u\to v}^{\beta} & \text{otherwise.} \end{cases}$$

Note that the consensus propagation iteration (5.10)–(5.11) is equivalent to

$$\begin{aligned} K_{i\to j}^{(t+1)} &= \mathcal{F}_{i\to j}(K^{(t),i\to j}), \\ \mu_{i\to j}^{(t+1)} &= \mathcal{G}_{i\to j}(\mu^{(t),i\to j}, K^{(t),i\to j}), \end{aligned}$$

for $t \in T^{i\to j}$. Hence, the vectors $K^{(t),i\to j}$ and $\mu^{(t),i\to j}$ capture the relevant information known by vertex i when it computes an update for vertex j at time a $t \in T^{i\to j}$.

The following lemma is a variant of the "box condition" argument of Proposition 6.2.1 in [14].

Lemma 5.7 *For any $\delta > 0$,*

$$\limsup_{t\to\infty} \|\mu^{(t)} - \mu^{\beta}\|_{\infty} \leq \frac{1+\alpha}{1-\alpha}\delta.$$

Proof. Fix $\delta > 0$. By Lemma 5.6, pick ϵ_2 so that if $\|K - K^{\beta}\|_{\infty} < \epsilon_2$, then $\|\nu(K) - \mu^{\beta}\|_{\infty} < \delta$. By Lemma 5.4, define the time T_2 so that for all $t \geq T_2$, $\|K^{(t)} - K^{\beta}\|_{\infty} < \epsilon_2$.

Define the quantity

$$\Delta \triangleq \max_{T_2 \leq t \leq E(T_2)} \|\mu^{(t)} - \mu^{\beta}\|_{\infty}.$$

For $k \geq 0$, and $(i,j) \in \vec{E}$, define $\mathcal{A}_{k,i\to j} \subset \mathbb{R}$ to be the set of real numbers $\mu_{i\to j} \in \mathbb{R}$ such that

$$|\mu_{i\to j} - \mu^{\beta}| < \alpha^k \Delta + \frac{1+\alpha}{1-\alpha}\delta.$$

Define $\mathcal{A}_k \subset \mathbb{R}^{\vec{E}}$ to be the set of vectors $\mu \in \mathbb{R}^{\vec{E}}$ such that

$$\mu_{i\to j} \in \mathcal{A}_{k,i\to j}, \quad \forall\ (i,j) \in \vec{E}.$$

We would like to show that, for every $k \geq 0$, there exists a time $t_k \geq 0$ such that:

(a) For all $t \geq t_k$,

$$\mu^{(t)} \in \mathcal{A}_k. \tag{5.26}$$

(b) For all $(i,j) \in \vec{E}$ and $t \in T^{i\to j}$ with $t \geq t_k$,

$$\mu^{(t),i\to j} \in \mathcal{A}_k. \tag{5.27}$$

We proceed by induction on k:
(Base Step.) Set $t_0 = E(T_2)$. Clearly (5.26) and (5.27) hold at time $t = t_0$, by our assumptions. Assume that, for some $s \geq t_0$, they continue to hold for all times $t_0 \leq t \leq s$. Then, if $s \in T^{i\to j}$,

$$\begin{aligned}
|\mu_{i\to j}^{(s+1)} - \mu_{i\to j}^{\beta}| &\leq |\mu_{i\to j}^{(s+1)} - \nu(K^{(s),i\to j})_{i\to j}| + |\nu(K^{(s),i\to j})_{i\to j} - \mu_{i\to j}^{\beta}| \\
&\leq |\mathcal{G}_{ij}(\mu^{(s),i\to j}, K^{(s),i\to j}) - \mathcal{G}_{ij}(\nu(K^{(s),i\to j}), K^{(s),i\to j})| \\
&\quad + \|\nu(K^{(s),i\to j}) - \mu^{\beta}\|_\infty \\
&< \alpha\|\mu^{(s),i\to j} - \nu(K^{(s),i\to j})\|_\infty + \delta \\
&< \alpha\|\mu^{(s),i\to j} - \mu^{\beta}\|_\infty + (1+\alpha)\delta \\
&< \alpha\left(\Delta + \frac{1+\alpha}{1-\alpha}\delta\right) + (1+\alpha)\delta \\
&= \alpha\Delta + \frac{1+\alpha}{1-\alpha}\delta.
\end{aligned}$$

Thus, $\mu_{i\to j}^{(s+1)} \in \mathcal{A}_{1,i\to j}$. In $s \notin T^{i\to j}$, the quantity $\mu_{i\to j}^{(s+1)}$ is unchanged, the $\mu_{i\to j}^{(s+1)} \in \mathcal{A}_{0,i\to j}$. In any event, (5.26)–(5.27) continue to hold at time $s+1$.
(Induction Step.) Assume that t_k exists, for some $k \geq 0$. For $(i,j) \in \vec{E}$, define

$t^{i\to j}$ to be the first element of $T^{i\to j}$ with $t^{i\to j} \geq t_k$. Then, if $s \in T^{i\to j}$,

$$\begin{aligned}
|\mu_{i\to j}^{(s+1)} - \mu_{i\to j}^{\beta}| &\leq |\mu_{i\to j}^{(s+1)} - \nu(K^{(s),i\to j})_{i\to j}| + |\nu(K^{(s),i\to j})_{i\to j} - \mu_{i\to j}^{\beta}| \\
&\leq |\mathcal{G}_{ij}(\mu^{(s),i\to j}, K^{(s),i\to j}) - \mathcal{G}_{ij}(\nu(K^{(s),i\to j}), K^{(s),i\to j})| \\
&\quad + \|\nu(K^{(s),i\to j}) - \mu^{\beta}\|_{\infty} \\
&< \alpha\|\mu^{(s),i\to j} - \nu(K^{(s),i\to j})\|_{\infty} + \delta \\
&< \alpha\|\mu^{(s),i\to j} - \mu^{\beta}\|_{\infty} + (1+\alpha)\delta \\
&< \alpha\left(\alpha^k\Delta + \frac{1+\alpha}{1-\alpha}\delta\right) + (1+\alpha)\delta \\
&= \alpha^{k+1}\Delta + \frac{1+\alpha}{1-\alpha}\delta.
\end{aligned}$$

Since $\mu_{i\to j}^{(s+1)}$ is not updated for $s \notin T^{i\to j}$, we have $\mu_{i\to j}^{(s)} \in \mathcal{A}_{k+1,i\to j}$, for all $s \geq t^{i\to j}+1$. Thus, if we define $t_{k+1} \geq E(t^{i\to j}+1)$, for all $(i,j) \in \vec{E}$, (a) and (b) will hold for $k+1$.

We have established that

$$\limsup_{t\to\infty} \|\mu^{(t)} - \mu^{\beta}\|_{\infty} \leq \alpha^k\Delta + \frac{1+\alpha}{1-\alpha}\delta,$$

for all $k \geq 0$. Taking a limit as $k\to\infty$, we get the desired result. ■

Overall Convergence

We are now ready to prove Theorem 5.3.

Theorem 5.3 *The following hold:*

(i) There exist unique vectors $K^{\beta} \in \mathbb{R}_{+}^{\vec{E}}$ and $\mu^{\beta} \in \mathbb{R}^{\vec{E}}$ that satisfy

$$\begin{aligned}
K_{i\to j}^{\beta} &= \frac{1 + \sum_{u\in N(i)\setminus j} K_{u\to i}^{\beta}}{1 + \frac{1}{\beta Q_{ij}}\left(1 + \sum_{u\in N(i)\setminus j} K_{u\to i}^{\beta}\right)}, \\
\mu_{i\to j}^{\beta} &= \frac{y_i + \sum_{u\in N(i)\setminus j} K_{u\to i}^{\beta}\mu_{u\to i}^{\beta}}{1 + \sum_{u\in N(i)\setminus j} K_{u\to i}^{\beta}},
\end{aligned}$$

for all $i \in V$ and $j \in N(i)$.

(ii) Suppose that the message updates satisfy the total asynchronism condition of Assumption 2.1. Then, independent of the initial condition $(\mu^{(0)}, K^{(0)}) \in \mathbb{R}^{\vec{E}} \times \mathbb{R}_+^{\vec{E}}$,

$$\lim_{t\to\infty} K^{(t)} = K^\beta, \quad \textit{and} \quad \lim_{t\to\infty} \mu^{(t)} = \mu^\beta.$$

(iii) For every $i \in V$,

$$x_i^\beta = \frac{y_i + \sum_{u\in N(i)} K_{u\to i}^\beta \mu_{u\to i}^\beta}{1 + \sum_{u\in N(i)} K_{u\to i}^\beta}.$$

Hence, under the assumptions of Part (ii),

$$\lim_{t\to\infty} x^{(t)} = x^\beta.$$

Proof. Existence and uniqueness of the fixed point K^β and convergence of the vector $K^{(t)}$ to K^β follow from Lemmas 5.3 and 5.4, respectively. Existence and uniqueness of the fixed point μ^β follows from Lemma 5.6.

Part (ii) follows from Lemma 5.7.

Part (iii) follows from the fact that , when the min-sum algorithm converges for pairwise quadratic optimization problems, it computes the global minimum. [90, 79, 88, 60]. ■

5.7.2 Proof of Theorems 5.4 and 5.5

In this section, we will prove Theorems 5.4 and 5.5. We will start with some preliminary lemmas.

Preliminary Lemmas

The following lemma provides bounds on k^β and γ^β in terms of β.

Lemma 5.8 *If* $d = 2$,

$$2\sqrt{\beta} - 1/2 < k^\beta < 2\sqrt{\beta},$$
$$\frac{1}{2\sqrt{\beta}+1} < \gamma^\beta < \frac{1}{2\sqrt{\beta}+1/2}.$$

If $d > 2$,

$$\left(1 - \frac{1}{d-1}\right)\beta - \frac{1}{d-1} < k^\beta < \beta,$$
$$\frac{1}{1+(d-1)\beta} < \gamma^\beta < \frac{1}{(d-2)\beta}.$$

Proof. Starting with the fixed point equation (5.20), some algebra leads to

$$\frac{d-1}{\beta}(k^\beta)^2 + (2 + \frac{1}{\beta} - d)k^\beta - 1 = 0.$$

The quadratic formula gives us

$$k^\beta = \frac{\beta}{2} - \frac{\beta+1}{2(d-1)} + \sqrt{\left(\frac{\beta}{2} - \frac{\beta+1}{2(d-1)}\right)^2 + 4\frac{\beta}{d-1}},$$

from which it is easy to derived the desired bounds. ■

The following lemma offers useful expressions for the fixed point μ^β and the optimal solution x^β.

Lemma 5.9

$$\mu^\beta = \sum_{\tau=0}^{\infty} \gamma^\beta (1-\gamma^\beta)^\tau \hat{P}^\tau \hat{y}, \tag{5.28a}$$

$$x^\beta = \frac{y}{1+dk^\beta} + \frac{dk^\beta}{1+dk^\beta} \sum_{\tau=0}^{\infty} \gamma^\beta (1-\gamma^\beta)^\tau A\hat{P}^\tau \hat{y}. \tag{5.28b}$$

Proof. If we consider the algorithm when $k_0 = k^\beta$, then $k_t = k^\beta$ and $\gamma_t = \gamma^\beta$ for all $t \geq 0$. Then, using (5.22) and induction, we have

$$\mu^{(t)} = \sum_{\tau=0}^{t} \gamma^\beta (1-\gamma^\beta)^\tau \hat{P}^\tau \hat{y},$$
$$x^{(t)} = \frac{y}{1+dk^\beta} + \frac{dk^\beta}{1+dk^\beta} \sum_{\tau=0}^{t} \gamma^\beta (1-\gamma^\beta)^\tau A\hat{P}^\tau \hat{y}.$$

The result follows from the fact that, as $t \to \infty$, $\mu^{(t)} \to \mu^\beta$ and $x^{(t)} \to x^\beta$ (Theorem 5.3). ■

The following lemma provides an estimate of the distance between fixed points μ^β and $\mu^{\beta'}$ in terms of the quantity $|\gamma^\beta - \gamma^{\beta'}|$.

Lemma 5.10 *Given $0 \le \beta' < \beta$, we have*

$$\|\mu^\beta - \mu^{\beta'}\|_{2,nd} \le \tau^\star(\gamma^{\beta'} - \gamma^\beta)(1 + 4/\gamma^\beta).$$

Proof. Using (5.28),

$$\begin{aligned}
&\|\mu^\beta - \mu^{\beta'}\|_{2,nd} \\
&= \left\| \sum_{\tau=0}^{\infty} \gamma^\beta (1-\gamma^\beta)^\tau \hat{P}^\tau \hat{y} - \sum_{\tau=0}^{\infty} \gamma^{\beta'} (1-\gamma^{\beta'})^\tau \hat{P}^\tau \hat{y} \right\|_{2,nd} \\
&\le \left\| \sum_{\tau=0}^{\infty} \Big(\gamma^\beta (1-\gamma^\beta)^\tau - \gamma^{\beta'} (1-\gamma^{\beta'})^\tau \Big) \hat{P}^\tau \right\|_{2,nd}.
\end{aligned}$$

Since

$$\sum_{\tau=0}^{\infty} \Big(\gamma^\beta (1-\gamma^\beta)^\tau - \gamma^{\beta'} (1-\gamma^{\beta'})^\tau \Big) = 0,$$

we have

$$\begin{aligned}
&\|\mu^\beta - \mu^{\beta'}\|_{2,nd} \\
&\le \left\| \sum_{\tau=0}^{\infty} \Big(\gamma^\beta (1-\gamma^\beta)^\tau - \gamma^{\beta'} (1-\gamma^{\beta'})^\tau \Big) (\hat{P}^\tau - \hat{P}^\star) \right\|_{2,nd} \\
&= \left\| \sum_{\tau=0}^{\infty} \sum_{s=\tau}^{\infty} \Big((\gamma^\beta)^2 (1-\gamma^\beta)^s \right. \\
&\qquad \left. - \ (\gamma^{\beta'})^2 (1-\gamma^{\beta'})^s \Big) (\hat{P}^\tau - \hat{P}^\star) \right\|_{2,nd} \\
&= \left\| \sum_{s=0}^{\infty} \Big((\gamma^\beta)^2 (1-\gamma^\beta)^s \right. \\
&\qquad \left. - \ (\gamma^{\beta'})^2 (1-\gamma^{\beta'})^s \Big) \sum_{\tau=0}^{s} (\hat{P}^\tau - \hat{P}^\star) \right\|_{2,nd} \\
&\le \tau^\star \sum_{s=0}^{\infty} |(\gamma^\beta)^2 (1-\gamma^\beta)^s - (\gamma^{\beta'})^2 (1-\gamma^{\beta'})^s|.
\end{aligned}$$

Hence, we wish to bound the sum

$$\Delta \triangleq \sum_{s=0}^{\infty} |(\gamma^{\beta})^2(1-\gamma^{\beta})^s - (\gamma^{\beta'})^2(1-\gamma^{\beta'})^s|.$$

Set

$$T \triangleq \left\lfloor 2\frac{\log\gamma^{\beta'} - \log\gamma^{\beta}}{\log(1-\gamma^{\beta}) - \log(1-\gamma^{\beta'})} \right\rfloor.$$

Note that

$$\begin{aligned}
(\gamma^{\beta})^2(1-\gamma^{\beta})^s &\le (\gamma^{\beta'})^2(1-\gamma^{\beta'})^s, && \text{if } s \le T,\\
(\gamma^{\beta})^2(1-\gamma^{\beta})^s &\ge (\gamma^{\beta'})^2(1-\gamma^{\beta'})^s, && \text{if } s > T.
\end{aligned}$$

Holding γ^{β} fixed, it is easy to verify that T is nondecreasing as $\gamma^{\beta'} \searrow \gamma^{\beta}$. Hence,

$$\begin{aligned}
T &\le 2\frac{\log\gamma^{\beta'} - \log\gamma^{\beta}}{\log(1-\gamma^{\beta}) - \log(1-\gamma^{\beta'})}\\
&\le \lim_{\gamma^{\beta'}\searrow\gamma^{\beta}} 2\frac{\log\gamma^{\beta'} - \log\gamma^{\beta}}{\log(1-\gamma^{\beta}) - \log(1-\gamma^{\beta'})}\\
&= 2(1-\gamma^{\beta})/\gamma^{\beta}.
\end{aligned} \tag{5.29}$$

Using the above results,

$$\begin{aligned}
\Delta &= \sum_{s=0}^{T} \left((\gamma^{\beta'})^2(1-\gamma^{\beta'})^s - (\gamma^{\beta})^2(1-\gamma^{\beta})^s\right)\\
&\quad + \sum_{s=T+1}^{\infty} \left((\gamma^{\beta})^2(1-\gamma^{\beta})^s - (\gamma^{\beta'})^2(1-\gamma^{\beta'})^s\right)\\
&= \gamma^{\beta'} - \gamma^{\beta} - 2\gamma^{\beta'}(1-\gamma^{\beta'})^{T+1} + 2\gamma^{\beta}(1-\gamma^{\beta})^{T+1}\\
&\le \gamma^{\beta'} - \gamma^{\beta} + 2\gamma^{\beta'}\left((1-\gamma^{\beta})^{T+1} - (1-\gamma^{\beta'})^{T+1}\right).
\end{aligned}$$

Now, note that if $0 < a \le b \le 1$, for integer $\ell > 0$,

$$\begin{aligned}
b^{\ell} - a^{\ell} &= b^{\ell}(1-(a/b)^{\ell}) = b^{\ell}(1-a/b)\sum_{i=0}^{\ell-1}(a/b)^i\\
&\le \ell b^{\ell-1}(b-a) \le \ell(b-a).
\end{aligned}$$

Applying this inequality and using (5.29), we have

$$\begin{aligned}\Delta &\leq (\gamma^{\beta'} - \gamma^{\beta})\left(1 + 2(T+1)\gamma^{\beta'}\right) \\ &\leq (\gamma^{\beta'} - \gamma^{\beta})\left(1 + 2\gamma^{\beta'}(2/\gamma^{\beta} - 1)\right) \\ &\leq (\gamma^{\beta'} - \gamma^{\beta})(1 + 4\gamma^{\beta'}/\gamma^{\beta}) \\ &\leq (\gamma^{\beta'} - \gamma^{\beta})(1 + 4/\gamma^{\beta}),\end{aligned}$$

which completes the proof. ■

The following lemma characterizes the rate at which $\gamma_t \searrow \gamma^{\beta}$.

Lemma 5.11 *Assume that $\gamma^{\beta} \leq \gamma_0 \leq 1$. Then, $\{\gamma_t\}$ is a nonincreasing sequence and*

$$|\gamma_t - \gamma^{\beta}| \leq \frac{(d-1)^t}{(1/\beta + \gamma^{\beta} + d - 1)^{2t}}.$$

Proof. Define the function

$$f(\gamma) \triangleq \frac{1}{1 + \frac{d-1}{1/\beta + \gamma}}.$$

Note that, from the definition of γ_t and (5.21), $\gamma_t = f(\gamma_{t-1})$. Further, from the definition of γ^{β} and (5.20), it is clear that $\gamma^{\beta} = f(\gamma^{\beta})$. Since $k_0 \leq k^{\beta}$, then $\gamma_0 \geq \gamma^{\beta}$, and since $k_t \nearrow k^{\beta}$ (from Lemma 5.1(ii)), $\gamma_t \searrow \gamma^{\beta}$. Also, if $\gamma \in [\gamma^{\beta}, 1]$,

$$f'(\gamma) = \frac{d-1}{(1/\beta + \gamma + d - 1)^2} \leq \frac{d-1}{(1/\beta + \gamma^{\beta} + d - 1)^2}.$$

Then, by the Mean Value Theorem,

$$\begin{aligned}|\gamma_t - \gamma^{\beta}| &= |f(\gamma_{t-1}) - f(\gamma^{\beta})| \\ &\leq \max_{\gamma \in [\gamma^{\beta}, 1]} |f'(\gamma)||\gamma_{t-1} - \gamma^{\beta}| \\ &\leq \frac{d-1}{(1/\beta + \gamma^{\beta} + d - 1)^2}|\gamma_{t-1} - \gamma^{\beta}| \\ &\leq \frac{(d-1)^t}{(1/\beta + \gamma^{\beta} + d - 1)^{2t}}|\gamma_0 - \gamma^{\beta}|.\end{aligned}$$

■

The following lemma establishes a bound on the distance between $x^{(t)}$ and $\bar{y}\mathbf{1}$ in terms of the distance between $\mu^{(t)}$ and μ^{β}.

Lemma 5.12

$$\|x^{(t)} - \bar{y}\mathbf{1}\|_{2,n} \leq \gamma_t + \gamma^{\beta}\tau^{\star} + \|\mu^{(t)} - \mu^{\beta}\|_{2,nd}.$$

Proof. First, note that, using (5.28),

$$\begin{aligned}
\|\mu^{\beta} - \hat{P}^{\star}\hat{y}\|_{2,nd} &= \left\|\sum_{\tau=0}^{\infty}\gamma^{\beta}(1-\gamma^{\beta})^{\tau}\hat{P}^{\tau} - \hat{P}^{\star}\right\|_{2,nd} \\
&= \left\|\sum_{\tau=0}^{\infty}\gamma^{\beta}(1-\gamma^{\beta})^{\tau}(\hat{P}^{\tau} - \hat{P}^{\star})\right\|_{2,nd} \\
&= \left\|\sum_{\tau=0}^{\infty}(\gamma^{\beta})^{2}\sum_{s=\tau}^{\infty}(1-\gamma^{\beta})^{s}(\hat{P}^{\tau} - \hat{P}^{\star})\right\|_{2,nd} \\
&\leq (\gamma^{\beta})^{2}\sum_{s=0}^{\infty}(1-\gamma^{\beta})^{s}\left\|\sum_{\tau=0}^{s}(\hat{P}^{\tau} - \hat{P}^{\star})\right\|_{2,nd} \\
&\leq \gamma^{\beta}\tau^{\star}.
\end{aligned} \tag{5.30}$$

Next, using Theorem 5.1, Lemma 5.8, and (5.30), we have

$$\begin{aligned}
\bar{y}\mathbf{1} &= \lim_{\beta\to\infty} x^{\beta} \\
&= \lim_{\beta\to\infty} \frac{y}{1+dk^{\beta}} + \frac{dk^{\beta}}{1+dk^{\beta}}A\mu^{\beta} \\
&= \lim_{\beta\to\infty} A\mu^{\beta} \\
&= A\hat{P}^{\star}\hat{y}.
\end{aligned}$$

Now,

$$\begin{aligned}
\|x^{(t)} - \bar{y}\mathbf{1}\|_{2,n} &\leq \frac{1}{1+dk_t}\|y - \bar{y}\mathbf{1}\|_{2,n} \\
&\quad + \frac{dk_t}{1+dk_t}\|A\mu^{(t)} - \bar{y}\mathbf{1}\|_{2,n} \\
&\leq \gamma_t + \|A\mu^{(t)} - \bar{y}\mathbf{1}\|_{2,n} \\
&\leq \gamma_t + \|A\mu^{(t)} - A\hat{P}^{\star}\hat{y}\|_{2,n}
\end{aligned}$$

By examining the structure of A, it follows from the Cauchy-Schwartz Inequality that

$$\|A(\mu^{(t)} - \hat{P}^\star \hat{y})\|_{2,n} \leq \|\mu^{(t)} - \hat{P}^\star \hat{y}\|_{2,nd}.$$

Thus, using (5.30)

$$\begin{aligned}\|x^{(t)} - \bar{y}\mathbf{1}\|_{2,n} &\leq \gamma_t + \|\mu^{(t)} - \hat{P}^\star \hat{y}\|_{2,nd} \\ &\leq \gamma_t + \|\mu^\beta - \hat{P}^\star \hat{y}\|_{2,nd} + \|\mu^{(t)} - \mu^\beta\|_{2,nd} \\ &\leq \gamma_t + \gamma^\beta \tau^\star + \|\mu^{(t)} - \mu^\beta\|_{2,nd}.\end{aligned}$$

■

Proof of Theorem 5.4

Theorem 5.4 follows immediately from the following lemma.

Lemma 5.13 *Fix $\epsilon > 0$, and pick β so that*

$$\begin{aligned}\beta &\geq \max\left\{(2(1+\tau^\star)/\epsilon - 1/2)^2/4, 9/16\right\}, && \text{if } d = 2, \\ \beta &\geq \max\left\{2(1+\tau^\star)/(\epsilon(d-2)), 3/(d-2)\right\}, && \text{if } d > 2.\end{aligned}$$

Assume that $k_0 \leq k^\beta$. Define

$$t^* \triangleq \left(1 + 2\sqrt{\beta}\right) \log\left(\frac{2 + 9\tau^\star\left(5 + 8\sqrt{\beta}\right)\left(1/2 + \sqrt{\beta}\right)}{\epsilon/2}\right),$$

if $d = 2$, and

$$t^* \triangleq (1 + (d-1)\beta) \log\left(\frac{2 + 4\tau^\star\left(5 + 4(d-1)\beta\right)}{\epsilon/2}\right),$$

if $d > 2$. Then, t^* is an ϵ-convergence time.

Proof. Let β_t be the value of β implied by k_t, that is, the unique value such that $k_t = k^{\beta_t}$. Define

$$\Delta_t \triangleq \|\mu^{(t)} - \mu^{\beta_t}\|_{2,nd}.$$

Note that the matrix $\hat{P}$ is doubly stochastic and hence nonexpansive under the $\|\cdot\|_{2,nd}$ norm. Then, from (5.22) and the fact that μ^{β_t} is a fixed point,

$$\begin{aligned}
\Delta_t &= \|\gamma_t \hat{y} + (1-\gamma_t)\hat{P}\mu^{(t-1)} - \gamma_t \hat{y} - (1-\gamma_t)\hat{P}\mu^{\beta_t}\|_{2,nd} \\
&= \|(1-\gamma_t)\hat{P}(\mu^{(t-1)} - \mu^{\beta_t})\|_{2,nd} \\
&\leq (1-\gamma_t)\|\mu^{(t-1)} - \mu^{\beta_t}\|_{2,nd} \\
&\leq (1-\gamma^\beta)\|\mu^{(t-1)} - \mu^{\beta_t}\|_{2,nd} \\
&\leq (1-\gamma^\beta)\left(\Delta_{t-1} + \|\mu^{\beta_{t-1}} - \mu^{\beta_t}\|_{2,nd}\right).
\end{aligned}$$

Now, using Lemmas 5.10 and 5.11,

$$\begin{aligned}
\Delta_t &\leq (1-\gamma^\beta)\left(\Delta_{t-1} + \tau^*(\gamma_{t-1} - \gamma_t)(1 + 4/\gamma_t)\right) \\
&\leq (1-\gamma^\beta)\left(\Delta_{t-1} + \tau^*\left(\gamma_{t-1} - \gamma^\beta\right)\left(1 + 4/\gamma^\beta\right)\right) \\
&\leq (1-\gamma^\beta)\left(\Delta_{t-1} + \tau^*\alpha^{t-1}\left(1 + 4/\gamma^\beta\right)\right).
\end{aligned} \tag{5.31}$$

Here, we define

$$\alpha \triangleq \begin{cases} 1/(\gamma^\beta + 1)^2, & \text{if } d = 2, \\ 1/(d-1), & \text{if } d > 2. \end{cases}$$

We would like to ensure that $\alpha < 1 - \gamma^\beta$. For $d = 2$, some algebra reveals that this is is true when $0 < \gamma^\beta < (\sqrt{5}-1)/2$. By the fact that $\beta \geq 9/16$ and Lemma 5.8, we have

$$0 < \gamma^\beta < \frac{1}{2\sqrt{\beta} + 1} \leq 2/5 < \frac{\sqrt{5}-1}{2}.$$

For $d > 2$, using the fact that $\beta \geq 3/(d-2)$ and Lemma 5.8,

$$\begin{aligned}
0 < \frac{\alpha}{1-\gamma^\beta} &< \frac{(d-2)\beta}{(d-1)((d-2)\beta - 1)} \\
&< \frac{3}{2(d-1)} \leq 3/4 < 1.
\end{aligned} \tag{5.32}$$

By induction using (5.31), we have

$$\begin{aligned}
\Delta_t &\leq (1-\gamma^\beta)^t + \tau^*\left(1 + 4/\gamma^\beta\right)\sum_{s=0}^{t-1}(1-\gamma^\beta)^{t-s}\alpha^s \\
&\leq (1-\gamma^\beta)^t\left(1 + \tau^*\frac{1 + 4/\gamma^\beta}{1 - \alpha/(1-\gamma^\beta)}\right).
\end{aligned}$$

Now, notice that using the above results and Lemmas 5.10, 5.11, and 5.12,

$$\begin{aligned}
&\|x^{(t)} - \bar{y}\mathbf{1}\|_{2,n} \\
&\leq \gamma_t + \gamma^\beta \tau^\star + \|\mu^{(t)} - \mu^\beta\|_{2,nd} \\
&\leq \gamma_t + \gamma^\beta \tau^\star + \Delta_t + \|\mu^{\beta_t} - \mu^\beta\|_{2,nd} \\
&\leq \gamma^\beta(1+\tau^\star) + (\gamma_t - \gamma^\beta) + \Delta_t \\
&\quad + \tau^\star(\gamma_t - \gamma^\beta)(1 + 4/\gamma^\beta) \\
&\leq \gamma^\beta(1+\tau^\star) + \alpha^t \\
&\quad + (1-\gamma^\beta)^t \left(1 + \tau^* \frac{1 + 4/\gamma^\beta}{1 - \alpha/(1-\gamma^\beta)}\right) \\
&\quad + \tau^\star \alpha^t (1 + 4/\gamma^\beta) \\
&\leq (1-\gamma^\beta)^t \left(2 + \tau^* \left(1 + 4/\gamma^\beta\right) \left(1 + \frac{1}{1 - \alpha/(1-\gamma^\beta)}\right)\right) \\
&\quad + \gamma^\beta(1+\tau^\star).
\end{aligned}$$

When $d = 2$, using Lemma 5.8 and the fact that $\beta \geq (2(1+\tau^\star)/\epsilon - 1/2)^2/4$, we have

$$(1+\tau^\star)\gamma^\beta < \frac{1+\tau^\star}{2\sqrt{\beta} + 1/2} \leq \epsilon/2.$$

Similarly, when $d > 2$, since $\beta \geq 2(1+\tau^\star)/(\epsilon(d-2))$,

$$(1+\tau^\star)\gamma^\beta < \frac{1+\tau^\star}{(d-2)\beta} \leq \epsilon/2.$$

Thus, we will have $\|x^{(t)} - \bar{y}\mathbf{1}\|_{2,n} \leq \epsilon$ if

$$(1-\gamma^\beta)^t \left(2 + \tau^* \left(1 + 4/\gamma^\beta\right) \left(1 + \frac{1}{1 - \alpha/(1-\gamma^\beta)}\right)\right) \leq \epsilon/2. \tag{5.33}$$

This will be true when

$$t \geq \frac{1}{\gamma^\beta} \log \left(\frac{2 + \tau^* \left(1 + 4/\gamma^\beta\right) \left(1 + \frac{1}{1 - \alpha/(1-\gamma^\beta)}\right)}{\epsilon/2} \right). \tag{5.34}$$

(We have used the fact that $\log(1-\gamma^\beta) \leq -\gamma^\beta$.) To complete the theorem, it suffices to show that t^* is an upper bound to the right hand side of (5.34).

Consider the $d = 2$ case. From Lemma 5.8, it follows that

$$1/\gamma^\beta < 1 + 2\sqrt{\beta},$$

$$1 + 4/\gamma^\beta < 5 + 8\sqrt{\beta}.$$

Finally,

$$\begin{aligned}\frac{1}{1-\alpha/(1-\gamma^\beta)} &= \frac{1}{1-\frac{1}{(1+\gamma^\beta)^2(1-\gamma^\beta)}} \\ &= \frac{1}{\gamma^\beta}\frac{(1+\gamma^\beta)^2(1-\gamma^\beta)}{1-\gamma^\beta-(\gamma^\beta)^2} \\ &= \frac{h(\gamma^\beta)}{\gamma^\beta}.\end{aligned}$$

Since $\beta \geq 9/16$, from Lemma 5.8, $\gamma^\beta \in (0, 1/2)$. It is easy to verify that for such γ^β, the rational function $h(\gamma^\beta)$ satisfies $h(\gamma^\beta) < h(1/2) = 9/2$. Thus,

$$\frac{1}{1-\alpha/(1-\gamma^\beta)} < \frac{9}{2\gamma^\beta} < 9/2 + 9\sqrt{\beta}.$$

For the $d > 2$ case, from Lemma 5.8, it follows that

$$1/\gamma^\beta < 1 + (d-1)\beta,$$

$$1 + 4/\gamma^\beta \leq 5 + 4(d-1)\beta.$$

Finally, using (5.32)

$$\frac{1}{1-\alpha/(1-\gamma^\beta)} < \frac{1}{1-3/4} = 4.$$

■

Proof of Theorem 5.5

Theorem 5.5 follows immediately from the following lemma.

Lemma 5.14 *Fix $\epsilon > 0$, and pick β so that*

$$\begin{aligned} \beta &\geq (2(1+\tau^\star)/\epsilon - 1/2)^2/4, && \text{if } d = 2, \\ \beta &\geq 2(1+\tau^\star)/(\epsilon(d-2)), && \text{if } d > 2. \end{aligned}$$

Assume that $k_0 = k^\beta$, and define

$$t^* \triangleq \begin{cases} \left(1 + 2\sqrt{\beta}\right)\log(2/\epsilon), & \text{if } d = 2, \\ (1 + (d-1)\beta)\log(2/\epsilon), & \text{if } d > 2. \end{cases}$$

Then, t^* is an ϵ-convergence time.

Proof. Note that in this case, we have $k_t = k^\beta$ and $\gamma_t = \gamma^\beta$, for all $t \geq 0$. We will follow the same strategy as the proof of Lemma 5.13. Define

$$\Delta_t \triangleq \|\mu^{(t)} - \mu^\beta\|_{2,nd}.$$

Note that the matrix $\hat{P}$ is doubly stochastic and hence nonexpansive under the $\|\cdot\|_{2,nd}$ norm. Then, from (5.22) and the fact that μ^{β_t} is a fixed point,

$$\begin{aligned} \Delta_t &= \|\gamma^\beta \hat{y} + (1-\gamma^\beta)\hat{P}\mu^{(t-1)} - \gamma^\beta \hat{y} - (1-\gamma^\beta)\hat{P}\mu^\beta\|_{2,nd} \\ &= \|(1-\gamma^\beta)\hat{P}(\mu^{(t-1)} - \mu^\beta)\|_{2,nd} \\ &\leq (1-\gamma^\beta)\|\mu^{(t-1)} - \mu^\beta\|_{2,nd} \\ &= (1-\gamma^\beta)\Delta_{t-1} \\ &\leq (1-\gamma^\beta)^t, \end{aligned}$$

where the last step follows by induction.

Now, notice that, using the result and Lemma 5.12,

$$\begin{aligned} \|x^{(t)} - \bar{y}\mathbf{1}\|_{2,n} &\leq \gamma^\beta(1+\tau^\star) + \Delta_t \\ &\leq \gamma^\beta(1+\tau^\star) + (1-\gamma^\beta)^t. \end{aligned}$$

When $d = 2$, using Lemma 5.8 and the fact that $\beta \geq (2(1+\tau^\star)/\epsilon - 1/2)^2/4$, we have

$$(1+\tau^\star)\gamma^\beta < \frac{1+\tau^\star}{2\sqrt{\beta}+1/2} \leq \epsilon/2.$$

Similarly, when $d > 2$, since $\beta \geq 2(1+\tau^\star)/(\epsilon(d-2))$,

$$(1+\tau^\star)\gamma^\beta < \frac{1+\tau^\star}{(d-2)\beta} \leq \epsilon/2.$$

Thus, we will have $\|x^{(t)} - \bar{y}\mathbf{1}\|_{2,n} \leq \epsilon$ if

$$(1-\gamma^\beta)^t \leq \epsilon/2.$$

This will be true when

$$t \geq \frac{1}{\gamma^\beta}\log(2/\epsilon). \tag{5.35}$$

(We have used the fact that $\log(1-\gamma^\beta) \leq -\gamma^\beta$.) To complete the theorem, it suffices to show that t^* is an upper bound to the right hand side of (5.35).

Consider the $d = 2$ case. From Lemma 5.8, it follows that

$$1/\gamma^\beta < 1 + 2\sqrt{\beta}.$$

For the $d > 2$ case, from Lemma 5.8, it follows that

$$1/\gamma^\beta < 1 + (d-1)\beta.$$

■

5.7.3 Proof of Theorem 5.6

Theorem 5.6 *For the cycle with n vertices, $\tau^\star \leq n/\sqrt{2}$.*

Proof. Let $e^{ij} \in \mathbb{R}^{2n}$ be the vector with the (i,j)th component equal to 1 and every other component equal to 0. It is easy to see that for any $(i,j) \in \vec{E}$,

$$\begin{aligned}\sup_t \left\| \sum_{\tau=0}^{t} (\hat{P}^\tau - \hat{P}^\star) e^{ij} \right\|_{2,2n}^2 &= \left\| \sum_{\tau=0}^{\lfloor n/2 \rfloor} (\hat{P}^\tau - \hat{P}^\star) e^{ij} \right\|_{2,2n}^2 \\ &\leq \frac{1}{2\sqrt{2}}.\end{aligned}$$

We then have

$$\begin{aligned}
\tau^\star &= \sup_{t,\mu} \frac{\left\|\sum_{\tau=0}^{t}(\hat{P}^\tau - \hat{P}^\star)\mu\right\|_{2,2n}}{\|\mu\|_{2,2n}} \\
&= \sup_{t,\mu} \frac{\left\|\sum_{\tau=0}^{t}(\hat{P}^\tau - \hat{P}^\star)\sum_{\{i,j\}}\mu_{ij}e^{ij}\right\|_{2,2n}}{\left\|\sum_{\{i,j\}}\mu_{ij}e^{ij}\right\|_{2,2n}} \\
&\leq \sup_{t,\mu} \frac{\sum_{\{i,j\}}\mu_{ij}\left\|\sum_{\tau=0}^{t}(\hat{P}^\tau - \hat{P}^\star)e^{ij}\right\|_{2,2n}}{\left\|\sum_{\{i,j\}}\mu_{ij}e^{ij}\right\|_{2,2n}} \\
&\leq \sup_{\mu} \frac{\sum_{\{i,j\}}\mu_{ij}}{2\sqrt{2}\|\sum_{\{i,j\}}\mu_{ij}e^{ij}\|_{2,2n}} \\
&= \sup_{\mu} \frac{\sum_{\{i,j\}}\mu_{ij}}{2\sqrt{2}\sqrt{\sum_{\{i,j\}}\mu_{ij}^2/2n}} \\
&\leq \sup_{\mu} \frac{\sum_{\{i,j\}}\mu_{ij}}{2\sqrt{2}\sum_{\{i,j\}}|\mu_{ij}|/2n} \\
&\leq \frac{n}{\sqrt{2}}.
\end{aligned}$$

■

6

CONCLUDING REMARKS

Graphical models encompass a class of optimization programs that are very relevant to modern engineering and management applications. Message-passing algorithms are an emerging set of methods to the seek to exploit graphical structure in order to provide efficient, decentralized solutions to large-scale optimization problems. They have shown much promise in practice, but a general theory has been thus far lacking.

In this thesis, we have sought to understand message-passing algorithms by highlighting the connections to classical methods in optimization. In particular, we have seen that messages, for resource allocation problems, can be interpreted as generalizations of prices or Lagrange multipliers, and that these methods yield equivalent solutions for convex problems. In the context of unconstrained convex optimization, we have developed sufficient conditions for the asynchronous convergence of message-passing algorithms that are equivalent to conditions known for the convergence of traditional local search methods, such as gradient descent or coordinate descent.

We have further argued that message-passing offers a number of advantages over traditional methods. For nonconvex resource allocation problems, message-passing algorithms can yield better solutions than price-based methods, by providing a richer description of the externalities of decentralized decision-making. In the case of the distributed consensus problem, we have shown that message-passing scales to large problem sizes more efficiently than competing local search methods.

However, there are many open questions that remain to be addressed. We highlight some of these below:

Stronger guarantees of solution quality. As we have seen in Theorems 3.4, 4.1, 4.2, and 5.3, message-passing fixed-points typically yield optimal solutions in the case of convex problems. For nonconvex problems, however, only much weaker guarantees of the form provided by Theorem 3.2 are available. However, as we saw in the example of Section 3.6.1, message-passing often has more impressive performance in practice than this theory suggests. It would be interesting to reconcile theory and empirical observation of message-passing as an approximation method for nonconvex problems. This could be done by, for example, identifying instances where global approximation guarantees can be proved.

Guarantees of convergence for a broader class of problems. In Chapter 4, we established convergence of message-passing for a scaled diagonally dominant class of convex programs under a totally asynchronous model of computation. A partially asynchronous or even synchronous model of computation may allow for convergence across a much broader class of convex problems, however, as is the case with local search methods.

More general analysis of rate of convergence. The analysis of the rate of convergence of message-passing algorithms is limited to special cases such as in Section 5.5. A more general understanding is needed, both to compare message-passing to other decentralized algorithms, such as local search methods, and to centralized algorithms, such as Newton's method.

Approximate parametrization of messages. When the decision variables take values over continuous domains, messages take the form of real-valued functions and cannot, in general, be computed or stored on digital computers. Implementable variants of message-passing, such as those proposed in Section 4.4, use finitely parameterized functions to approximate messages. Developing a theory for such approximate message-passing algorithms is crucial for any practical use. Here, ideas from the theory of approximate dynamic programming may prove relevant.

BIBLIOGRAPHY

[1] S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46:325–343, 2000.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(9):102–114, August 2002.

[3] D. Aldous. The $\zeta(2)$ limit in the random assignment problem. *Random Structures and Algorithms*, 18:381–418, 2001.

[4] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12:308–340, 1991.

[5] K. J. Arrow and L. Hurwicz, editors. *Studies in Resource Allocation*. Cambridge University Press, Cambridge, UK, 1977.

[6] E. Aurell, U. Gordon, and S. Kirkpatrick. Comparing beliefs, surveys, and random walks. In *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2005. MIT Press.

[7] Z. Bar-Yossef, T. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *Proceedings of the Annual Workshop on Randomization and Approximation Techniques (RANDOM)*, 2002.

[8] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani. Estimating aggregates on a peer-to-peer network. Technical report, Computer Science Department, Stanford University, 2003.

[9] M. Bayati, D. Shah, and M. Sharma. Maximum weight matching via max-product belief propagation. In *International Symposium of Information Theory*, Adelaide, Australia, September 2005.

[10] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara. Soft-output decoding algorithms in iterative decoding of turbo codes. *JPL TDA Progress Report*, 42(124):63–87, February 1996.

[11] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding. In *Proc. Int. Communications Conf.*, pages 1064–1070, Geneva, Switzerlang, May 1993.

[12] U. Bertelè and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, New York, 1972.

[13] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, 1995.

[14] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, Belmont, MA, 1997.

[15] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis. Convergence in multiagent coordination, consensus, and flocking. In *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference*, Seville, Spain, September 2005.

[16] H. L. Bodlaender. Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *J. Algorithms*, 11:631–643, 1990.

[17] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–21, 1993.

[18] S. Boyd, P. Diaconis, P. Parillo, and L. Xiao. Symmetry analysis of reversible Markov chains. *Internet Mathematics*, 2(1):31–71, 2005.

[19] S. Boyd, P. Diaconis, J. Sun, and L. Xiao. Fastest mixing Markov chain on a path. *The American Mathematical Monthly*, 113(1):70–74, January 2006.

[20] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, June 2006.

[21] A. Braunstein, M. Mézard, and R. Zecchina. Survey propagation: An algorithm for satisfiability. *Random Struct. Algorithms*, 27(2):201–226, 2005.

[22] A. Braunstein and R. Zecchina. Survey propagation as local equilibrium equations. *Journal of Statistical Mechanics: Theory and Experiment*, 2004(06):P06007, 2004.

[23] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, January 2007.

[24] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *International Conference on Data Engineering*, 2004.

[25] J. Coughlan and S. Ferreira. Finding deformable shapes using loopy belief propagation. In *European Conference on Computer Vision*, 2002.

[26] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7:279–301, 1989.

[27] P. Diaconis, S. Holmes, and R.M. Neal. Analysis of a nonreversible Markov chain sampler. *Annals of Applied Probability*, 10(3):726–752, 2000.

[28] R. Diekmann, A. Frommer, and B. Monien. Efficient schemes for nearest neighbor load balancing. *Parallel Computing*, 25(789–812), 1999.

[29] M. Fazel and M. Chiang. Network utility maximization with nonconcave utilities using sum-of-squares method. In *Proceedings of the 44th Conference on Decision and Control*, 2005.

[30] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.

[31] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *Intl. J. Computer Vision*, 40(1):25–47, 2000.

[32] W. T. Freeman and Y. Weiss. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47:736–744, 2001.

[33] A. Fréville. The multidimensional 0–1 knapsack problem: An overview. *European Journal of Operational Research*, 155:1–21, 2004.

[34] B. J. Frey. *Graphical Models for Machine Learning and Digital Communication*. M.I.T. Press, Cambridge, MA, 1998.

[35] R. G. Gallager. *Low-Density Parity Check Codes*. M.I.T. Press, Cambridge, MA, 1963.

[36] P. Hande, S. Zhang, and M. Chiang. Distributed rate allocation for inelastic flows. Submitted to IEEE/ACM Transactions of Networking, 2005.

[37] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, 2000.

[38] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[39] M. Jelasity and A. Montresor. Epedemic-style proactive aggregation in large overlay networks. In *Proceedings of the 24th International Conference on Distributed Computing*, 2004.

[40] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[41] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *ACM Symposium on Theory of Computing*, 2004.

[42] A. M. C. A. Koster, S. P. M. van Hoesel, and A. W. J. Kolen. Solving partial constraint satisfaction problems with tree decomposition. *Networks*, 40:170–180, 2002.

[43] S. J. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society Series B*, 50:157–224, 1988.

[44] S. L. Lauritzen. *Graphical Models*. Number 17 in Oxford Statistical Science Series. Clarendon Press, Oxford, UK, 1996.

[45] J. W. Lee, R. R. Mazumdar, and N. B. Shroff. Non-convex optimization and rate control for multi-class services in the Internet. *IEEE/ACM Trans. on Networking*, 13(4):841–853, 2005.

[46] M. Leone and A. Pagnani. Predicting protein functions with message passing algorithms. *Bioinformatics*, 21:239–247, 2005.

[47] S. Letovsky and S. Kasif. Predicting protein function from protein/protein interaction data: a probabilistic approach. *Bioinformatics*, 19 Suppl 1:i197–204, 2003.

[48] Q. Li and D. Rus. Global clock syncronization in sensor networks. *IEEE Transaction on Computers*, 55(2):214–226, 2006.

[49] J. Lin, A. S. Morse, and B. D. O. Anderson. The multi-agent rendezvous problem: an extended summary. In A. S. Morse, V. Kumar, and N. E. Leonard, editors, *Proceedings of the 2003 Block Island Workshop on Cooperative Control*, volume 309 of *Lecture Notes in Control and Information Sciences*, pages 257âĂŞ–282, New York, 2004. Springer Verlag.

[50] S. R. Madden, M. J. Franklin, J. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad hoc sensor networks. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*, 2002.

[51] S. R. Madden, R. Szewczyk, M. J. Franklin, and D. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, 2002.

[52] D. M. Malioutov, J. K. Johnson, and A. S. Willsky. Walk-sums and belief propagation in Gaussian graphical models. *Journal of Machine Learning Research*, 7:2031–2064, October 2006.

[53] E. N. Maneva, E. Mossel, and M. J. Wainwright. A new look at survey propagation and its generalizations. In *Proceedings of the Symposium on Discrete Algorithms*, 2005.

[54] A. McCallum. Efficiently inducing features of conditional random fields. In *Uncertainty in Artifificial Intelligence: Proceedings of the Nineteenth Conference*, 2003.

[55] M. Mézard, G. Parisi, and M. A. Virasoro. *Spin Glass Theory and Beyond*. World Scientific, Singapore, 1987.

[56] M. Mézard, G. Parisi, and R. Zecchina. Analytic and algorithmic solutions to random satisfiability problems. *Science*, 297(5582):812–815, 2002.

[57] C. C. Moallemi and B. Van Roy. Distributed optimization in adaptive networks. In *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.

[58] C. C. Moallemi and B. Van Roy. Consensus propagation. *IEEE Transactions on Information Theory*, 52(11):4753–4766, 2006.

[59] C. C. Moallemi and B. Van Roy. Consensus propagation. In *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2006. MIT Press.

[60] C. C. Moallemi and B. Van Roy. Convergence of the min-sum message passing algorithm for quadratic optimization. Technical report, Management Science & Engineering Department, Stanford University, 2006. URL: `http://moallemi.com/ciamac/papers/qms-2006.pdf`.

[61] C. C. Moallemi and B. Van Roy. Convergence of the min-sum algorithm for convex optimization. Technical report, Management Science & Engineering Department, Stanford University, 2007. URL: `http://moallemi.com/ciamac/papers/cc-2007.pdf`.

[62] C. C. Moallemi and B. Van Roy. A message-passing paradigm for resource allocation. Technical report, Management Science & Engineering Department, Stanford University, 2007. URL: `http://moallemi.com/ciamac/papers/ra-2007.pdf`.

[63] A. Montanari, B. Prabhakar, and D. Tse. Belief propagation based multiuser detection. In *Proceedings of the Allerton Conference on Communication, Control, and Computing*, 2005.

[64] A. Montresor, M. Jelasity, and O. Babaoglu. Robust aggregation protocols for large-scale overlay networks. In *Proceedings of the International Conference on Dependable Systems and Networks*, 2004.

[65] B. S. Mordukhovich. Nonlinear prices in nonconvex economies with classical Pareto and strong Pareto optimal allocations. *Positivity*, 9:541–568, 2005.

[66] L. Moreau. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions on Automatic Control*, 50(2):169–182, 2005.

[67] D. Mosk-Aoyama and D. Shah. Computing separable functions via gossip. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, 2006.

[68] S. Muthukrishnan, B. Ghosh, and M. Schultz. First and second order diffusive methods for rapid, coarse, distributed load balancing. *Theory of Computing Systems*, 31:331–354, 1998.

[69] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

[70] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufman, San Mateo, CA, 1988.

[71] D. Pinto, A. McCallum, X. Wei, and W. B. Croft. Table extraction using conditional random fields. In *Proceedings of the ACM SIGIR*, 2003.

[72] W. Ren and R. W. Beard. Consensus seeking in multiagent systems under dynamically changing interaction topologies. *IEEE Transaction on Automatic Control*, 50(5):655–661, 2005.

[73] T. Richardson and R. Urbanke. The capacity of low-density parity check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47:599–618, 2001.

[74] T. Richardson and R. Urbanke. An introduction to the analysis of iterative coding systems. In *Codes, Systems, and Graphical Models, IMA Volume in Mathematics and Its Applications*, pages 1–37. Springer, 2001.

[75] N. Robertson and P. D. Seymour. Graph minors. I. Excluding a forest. *J. Comb. Theory Series B*, 35:39–61, 1983.

[76] N. Robertson and P. D. Seymour. Graph minors. X. Obstructions to tree-decomposition. *J. Comb. Theory Series B*, 52:153–190, 1991.

[77] S. Roch. Bounding fastest mixing. *Electronic Communications in Probability*, 10:282–296, 2005.

[78] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, NJ, 1970.

[79] P. Rusmevichientong and B. Van Roy. An analysis of belief propagation on the turbo decoding graph with Gaussian densities. *IEEE Transactions on Information Theory*, 47(2):745–765, 2001.

[80] D. Shah. Max product for max-weight independent set and matching. Preprint. URL: `http://arxiv.org/abs/cs.DS/0508097`, August 2005.

[81] S. Shenker. Fundamental design issues for the future Internet. *IEEE Journal on Selected Areas in Communications*, 13(7):1176–1188, 1995.

[82] S. L. Smith, M. E. Broucke, , and B. A. Francis. A hierarchical cyclic pursuit scheme for vehicle networks. *Automatica*, 41(6):1045–1053, 2005.

[83] J. Sun, Shum H. Y, and N. N. Zheng. Stereo matching using belief propagation. In *European Conference on Computer Vision*, 2002.

[84] C. Sutton, K. Rohanimanesh, and A. McCallum. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proceedings of the International Conference for Machine Learning*, 2004.

[85] M. Talagrand. *Spin Glasses: A Challenge for Mathematicians*. Springer, 2003.

[86] S. Tatikonda and M. I. Jordan. Loopy belief propagation and Gibbs measures. In *Uncertainty in Artificial Intelligence: Proceedings of the Eighteenth Conference*, 2002.

[87] J. N. Tsitsiklis. *Problems in Decentralized Decision-Making and Computation*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1984.

[88] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, 49(5):1120–1146, 2003.

[89] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. *Statistics and Computing*, 14:143–166, 2004.

[90] Y. Weiss and W. T. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, 13:2173–2200, 2001.

[91] N. Wiberg. *Codes and Decoding on General Graphs*. PhD thesis, Linköping University, Linköping, Sweden, 1996.

[92] L. Xiao, S. Boyd, and S.-J. Kim. Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing*, 67(1):33–46, 2007.

[93] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *International Conference on Information Processing in Sensor Networks*, pages 63–70, Los Angeles, CA, April 2005.

[94] J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring wireless sensor networks. In *Proceedings of the International Workshop on Sensor Net Protocols and Applicatinos*, 2003.