On Distributed Optimization in Networked Systems

BJÖRN JOHANSSON

KTH Electrical Engineering

Doctoral Thesis in Telecommunication
Stockholm, Sweden 2008

KTH Electrical Engineering

On Distributed Optimization in Networked Systems

BJÖRN JOHANSSON

PhD Thesis
Stockholm, Sweden 2008

Automatic Control Laboratory
School of Electrical Engineering
Royal Institute of Technology (KTH)
SE-100 44 Stockholm
SWEDEN

TRITA-EE 2008:065
ISSN 1653-5146
ISBN 978-91-7415-190-9

Akademisk avhandling som med tillstånd av Kungliga Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie doktorsexamen fredagen den 30 januari 2009 klockan 13:15 i sal F3, Kungliga Tekniska högskolan, Lindstedtsvägen 26, Stockholm.

Tryck: Universitetsservice US-AB

Abstract

Numerous control and decision problems in networked systems can be posed as optimization problems. Examples include the framework of network utility maximization for resource allocation in communication networks, multi-agent coordination in robotics, and collaborative estimation in wireless sensor networks (WSNs). In contrast to classical distributed optimization, which focuses on improving computational efficiency and scalability, these new applications require simple mechanisms that can operate under limited communication. In this thesis, we develop several novel mechanisms for distributed optimization under communication constraints, and apply these to several challenging engineering problems.

In particular, we devise three tailored optimization algorithms relying only on nearest neighbor, also known as peer-to-peer, communication. Two of the algorithms are designed to minimize a non-smooth convex additive objective function, in which each term corresponds to a node in a network. The first method is an extension of the randomized incremental subgradient method where the update order is given by a random walk on the underlying communication graph, resulting in a randomized peer-to-peer algorithm with guaranteed convergence properties. The second method combines local subgradient iterations with consensus steps to average local update directions. The resulting optimization method can be executed in a peer-to-peer fashion and analyzed using epsilon-subgradient methods. The third algorithm is a center-free algorithm, which solves a non-smooth resource allocation problem with a separable additive convex objective function subject to a constant sum constraint.

Then we consider cross-layer optimization of communication networks, and demonstrate how optimization techniques allow us to engineer protocols that mimic the operation of distributed optimization algorithms to obtain an optimal resource allocation. We describe a novel use of decomposition methods for cross-layer optimization, and present a flowchart that can be used to categorize and visualize a large part of the current literature on this topic. In addition, we devise protocols that optimize the resource allocation in frequency-division multiple access (FDMA) networks and spatial reuse time-division multiple access (TDMA) networks, respectively.

Next we investigate some variants of the consensus problem for multi-robot coordination, for which it is usually standard to assume that agents should meet at the barycenter of the initial states. We propose a negotiation strategy to find an optimal meeting point in the sense that the agents' trajectories to the meeting point minimize a quadratic cost criterion. Furthermore, we also demonstrate how an augmented state vector can be used to boost the convergence rate of the standard linear distributed averaging iterations, and we present necessary and sufficient convergence conditions for a general version of these iterations.

Finally, we devise a generic optimization software component for WSNs. To this end, we implement some of the most promising optimization algorithms developed by ourselves and others in our WSN testbed, and present experimental results, which show that the proposed algorithms work surprisingly well.

To my mother, Inger, and my late father, Tommy.

Acknowledgments

"*Someone told me that each equation I included in the book would halve the sales.*"

S. W. Hawking, A Brief History of Time, 1988.[1]

Foremost, I would like to thank my advisor, Associate Professor Mikael Johansson, for your help, encouragement, and fruitful discussions; it has been a pleasure working with you! My co-advisor, Professor Karl Henrik Johansson, has also been most helpful. Everytime we spoke, both of you made me feel that I could conquer the world with my research. Not likely, I guess, but nevertheless an amazing feeling!

I'm grateful to all fellow PhD students in the control group and Karin Karlsson Eklund and Anneli Ström, for making KTH a fun and warm place to work.

A big thanks to all present and former collaborators: Cesare Maria Carretti, Assistant Professor Tamás Keviczky, Dr. Alberto Speranzon, Dr. Carlo Fischione, Tekn. Lic. Pablo Soldati, Dr. Constantin Adam, Professor Rolf Stadler, Associate Professor Mung Chiang, Assistant Professor Jianwei Huang, and Helen Li. Extra thanks to Cesare Maria Carretti who performed the NS2 simulations and did most of the coding of the WSN experiments in Chapter 6, and to Tekn. Lic. Pablo Soldati who performed the STDMA simulations in Chapter 4.

I am also very grateful to Associate Professor Mung Chiang for hosting my pleasant visit at Princeton University in 2007. At Princeton, I really enjoyed the lunches and discussions with Dr. Yung Yi.

Several free software packages, SeDuMi, YALMIP, TinyOS, Graphviz, and Jemdoc, as well as the websites `www.google.com` and `www.wikipedia.org`, have made my life substantially easier. Thanks.

Thanks to Dr. Märta Barenthin Syberg, Tekn. Lic. Oscar Flärdh, M.A. Dmitry Kiper, Tekn. Lic. Magnus Lindhé, Dr. Maben Rabi, and Tekn. Lic. Pablo Soldati for proofreading various parts of this thesis. The errors are of course my own fault. Thanks to PanGun Park, Piergiuseppe Di Marco, and Chithrupa Ramesh for assistance with motes and TinyOS.

Without financial support this work would not have been possible, and I'm grateful to the Swedish Research Council (CoopNets), the European Commission

[1] If we keep the potential circulation of this thesis modestly upper bounded by $6 \cdot 10^9$, then, since it contains over 100 equations, fewer than $4.7332 \cdot 10^{-21}$ persons will read it.

(Socrades), the Swedish Agency for Innovation Systems (WISA I/II), and ultimately the Swedish taxpayers for this support.

Finally, I would like to thank my family, friends, and especially the loves of my life, Selma and Charlotte, for being there.

Björn Lindgren Johansson
Stockholm, December 2008

Contents

Chapter 1

Introduction

> "*It follows that the Scientist, like the Pilgrim, must wend a straight and narrow path between the Pitfalls of Oversimplification and the Morass of Overcomplication.*"
>
> R. Bellman, Dynamic Programming, 1957.

The world is connected. Cellular phone networks, power grids, and the Internet are examples of vital infrastructures that connect everything on this planet. These crucial systems share one striking feature: they are composed of subsystems that make local decisions and coordinate with other subsystems to accomplish their tasks. We denote such systems as *networked systems*.

The most well-known networked system is probably the Internet, which comprises a mind boggling number of computers and users. It is interesting, and perhaps even surprising, that the Internet actually works as well as it does. Networked systems are also present where you might not suspect. For example, in most modern cars and aircraft, several controllers are working together to make your life as a driver or pilot easier and safer. A novel example is so-called wireless sensor networks (WSNs), which are composed of tiny wireless sensors, so-called *motes*. These sensor networks can be used for, e.g., monitoring and surveillance; see Fig. 1.1 for an example of a WSN. In addition, there are numerous examples of networked systems in nature as well, and the prototypical example is a school of fish, which is shown in Fig. 1.2.

We really depend on networked systems in our daily lives, but we do not yet understand these phenomena fully, and there is no comprehensive theory. Thus, engineers designing such large scale systems usually have to rely on intuition, trial and error, and empirical experience. Although these techniques are very useful, which we can deduce from the fact that many of these systems work extremely well, it is rather widely believed that we could do better if we had a firmer theoretical understanding of these phenomena. There is a major on-going research effort in understanding such systems, and this thesis is a part of that effort. Thus, the aim of this thesis is to add knowledge to the broad theory of networked systems.

Figure 1.1: The picture shows a part of the wireless sensor network testbed mounted in the ceiling at the Automatic Control Lab at KTH; see Maciel (2008) for full details. Each box contains a Tmote Sky mote equipped with a temperature sensor, and a majority of the motes also have an ultrasound receiver. The motes communicate with each other using radios compliant with the IEEE 802.15.4 standard. The wiring that can be seen supplies power and is not used for communication.

More specifically, in this thesis, we consider decision problems that are posed as optimization problems and where there is a natural communication structure imposed on the disparate parts of the system. The communication structure specifies which subsystems can directly communicate with each other; see Fig. 1.3 for an illustration. Two important problems emerge: *which* optimization problems should we solve and *how* should we solve them? We partly address both questions in this thesis. First, we consider the *how*-problem and devise specialized optimization algorithms, which only rely on nearest neighbor communication, or so-called *peer-to-peer* communication (Chapter 3). Second, we address the *which*-problem and consider resource allocation problems to obtain communication protocols that make communication networks more efficient (Chapter 4). Furthermore, we consider multi-agent rendezvous problems; we investigate where the agents should rendezvous and how they should get there (Chapter 5). Finally, we put our novel algorithms to the test, and we implement some of the most promising algorithms on our WSN testbed (Chapter 6).

Two applications in particular motivate this thesis: resource allocation in communication networks and wireless sensor networks.

1.1 Resource Allocation in Communication Networks

The world has an immense appetite for communication and bandwidth, and the demand seems to only increase. At the same time, the physical resources are limited, e.g., there is only a limited spectrum available for wireless communications.

Figure 1.2: A school of fish is a networked system; it is an example of rather simple subsystems cooperating to accomplish a task. In this case, the task is to form a school, which according to some theories create protection against predators (Cushing and Jones, 1968). Cooperation is an essential component in our notion of networked systems. Photography © Dejan Sarman | Dreamstime.com.

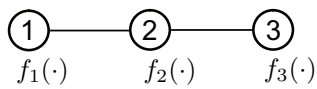


Figure 1.3: The figure shows an example of a networked system consisting of three nodes. A line between two nodes implies that the nodes can communicate with each other. Furthermore, each node is associated with a cost function, and the nodes cooperate to find the minimum of the sum of the cost functions, $\sum_{i=1}^{n} f_i(x)$. Numerous tasks in several applications can be posed as optimization problems of this type.

Therefore, there are financial incentives to use current and future communications hardware and the available medium as efficiently as possible.

A data network consists of several nodes (a node is, e.g., a cell phone) communicating with each other using some physical medium; see Fig. 1.4 for an illustration. The connections between nodes are called links, and they can be considered to be pipes in which data flow. If a node desires to communicate with a distant node, then it has to use several links to reach the other node.

A design strategy often used in engineering is to divide a system into separate subsystems (i.e., divide-and-conquer), which allows a more or less modular design. This paradigm has also been used in network design, and the resulting subsystems are often called layers. The layers are logically stacked on top of each other, where

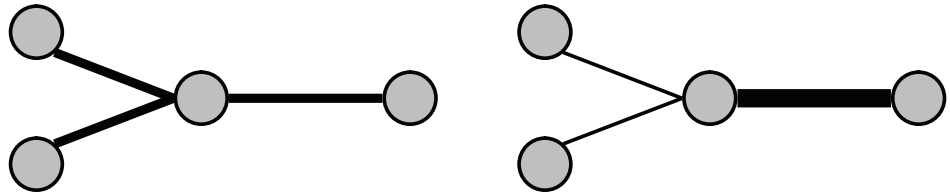


Figure 1.4: The circles denote nodes in a communication network with controllable link capacities, but the sum of the capacities is constant. The solid lines illustrate the links and the thickness of the lines corresponds to the capacity. The two nodes to the left and the node in the center desire to send as much data as possible to the rightmost node. (a) This is the network before cross-layer optimization. All links have the same capacity and the link to the right is a bottleneck. (b) This is the network after cross-layer optimization. Now, the two left links have less capacity and the right link has more capacity, and the bottleneck link has been relieved; more data can now be transferred from the left side and the center to the right side.

the lower layers provide basic functionality for the higher layers. The higher layers can use the functionality provided by the lower layers without knowing anything about their inner workings. The layered structure will make the design and the maintenance of the network design much simpler. However, if the layers are allowed to cooperate more closely, then there could be performance benefits. This cooperation can be achieved by using *cross-layer optimization*. The drawback with this increased entanglement is that the design probably has to be redone if one layer is changed. Thus, the performance is improved at the expense of greater complexity. An enlightening discussion on the potential benefits and pitfalls of cross-layer optimization can be found in Kawadia and Kumar (2005).

We consider communication networks where the physical medium is shared and the capacities can be changed by allocating some limited physical resource (e.g., a cellular phone network). Returning to the pipe analogy that we used earlier, we consider networks where the pipes have variable radii that we can control. More specifically, we will concentrate on the interplay between the data-flow rates and the capacity allocation in the links.

The benefit of cross-layer optimization is illustrated by the four node example network in Fig. 1.4. This network has links where the capacities are controllable, but the total capacity is limited. If capacity is shifted from lightly loaded links to heavily loaded links, the amount of data transferred over the network can increase.

Finally, an example of cross-layer optimization at work can be seen in Fig. 1.5. In this example, which is more thoroughly introduced and analyzed in Chapter 4, we consider a spatial reuse time-division multiple access communication network. Here is a brief description of how it works: in time-division multiple access networks, only one transmitter can transmit at a given time and the transmissions occur according

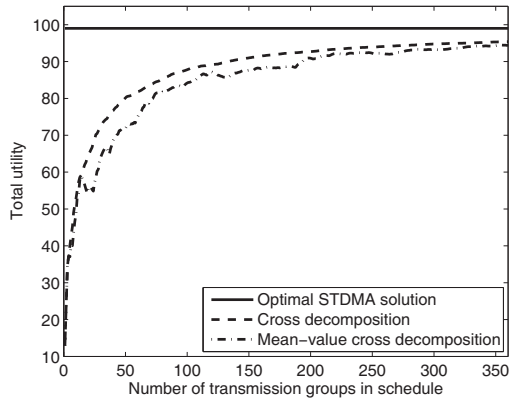


Figure 1.5: The figure shows the utility of a communication network versus the number of time slots in the transmission schedule. The solid line denotes the total utility (the utility should be as high as possible) using optimal Spatial reuse Time-Division Multiple Access (STDMA) of a communication network, which is computed using a centralized algorithm and provides an upper bound on the achievable performance. The dashed and dash dotted lines denote two algorithms, described and discussed in detail in Chapter 4, that computes a good STDMA schedule in a decentralized fashion.

to a schedule. In a *spatial reuse* time-division multiple access communication network, transmitters, which are sufficiently far away from each other, can transmit at the same time. This generalization increases the capacity of the network. But how can such a schedule be automatically constructed in a decentralized fashion? This is one of the topics in Chapter 4.

1.2 Wireless Sensor Networks

Wireless sensor networks are composed of tiny sensors with wireless communication abilities that cooperate to perform a task. An overview of potential applications and research challenges, can be found in, e.g., Culler et al. (2004) or Akyildiz et al. (2002). The motes are envisioned to be mass-produced at a very low cost and could therefore be deployed in huge amounts to allow ubiquitous sensing. Culler et al. (2004) suggest that the applications of WSNs can be roughly differentiated into monitoring space, monitoring things, and monitoring the interactions of things with each other and the encompassing space. Another way to partition sensor network applications, especially information processing, is into the following two areas: monitoring and estimation of spatially distributed and interacting phenomena, and increasing the fidelity and/or robustness of the estimation of a local phenomena by cooperation.

Wireless sensor networks is a very active area of research and there has been a lot of activity recently[1]. The author believes that many researchers are intrigued by this area due to the fundamental engineering challenges, on theoretical as well as practical levels. The challenges stem from inherent limitations such as energy scarcity; unreliable wireless communication links; limited processing and storage capabilities; and the distributed mode of operation.

An interesting example of a recent application involving monitoring space is a project in San Francisco, USA, where sensor networks are used to monitor parking spaces. The information will be used to provide real-time information on available spaces as well as to open up for popularity based parking fees (Markoff, 2008).

Another intriguing example is the Quake-catcher network (QCN, 2008), which uses built-in sensors in volunteers' laptops to detect earth-quakes. Such sensors are already mounted in most new laptops to detect if the laptop is dropped; if a sudden acceleration is detected, the hard drive is protected from impact.

Finally, in Fig. 1.6, a generic optimization example is shown. In this example, 6 motes cooperate to solve an optimization problem with a network-wide decision variable. The figure shows the convergence of a number of algorithms that exhibit strikingly different behavior; these algorithms and aspects of their implementation will be discussed in Chapter 6.

1.3 Why Should We Use Optimization?

Who can say "No" to something that is optimal? Optimization is an attractive framework to use for solving a multitude of decision problems; especially quantitative problems in engineering are well suited for this framework. The crux of this approach is that we need to summarize all design criteria into one performance measure. Once this major hurdle has been passed, we also need to actually find an optimal solution. Here we have a classic trade-off between fidelity and tractability: the closer we get to capturing all details of the engineering problem at hand, the more likely the resulting optimization problem will be very hard to solve. If the model is simple and yields an easily solvable optimization problem, then the problem may be pointless to solve since the solution does not say anything about the true system.

Optimization can enter into the system structure in several distinct ways. For example, when we consider resource allocation in communication networks, we interpret the communication network as an optimization solver, whereas when we consider WSNs, we devise a generic optimization software component that could be used for a variety of tasks. We will now elaborate on how optimization fits into the different applications, and we use the ideal feedback loop as a prototype. Further-

[1] A search for "sensor network" (searching for the whole phrase), November 14, 2008, on Google yields approximately $5.0 \cdot 10^6$ hits, a search on Google Scholar yields circa $7.0 \cdot 10^4$ hits, and a search on ISI Web of knowledge yields circa $1.8 \cdot 10^4$ hits.

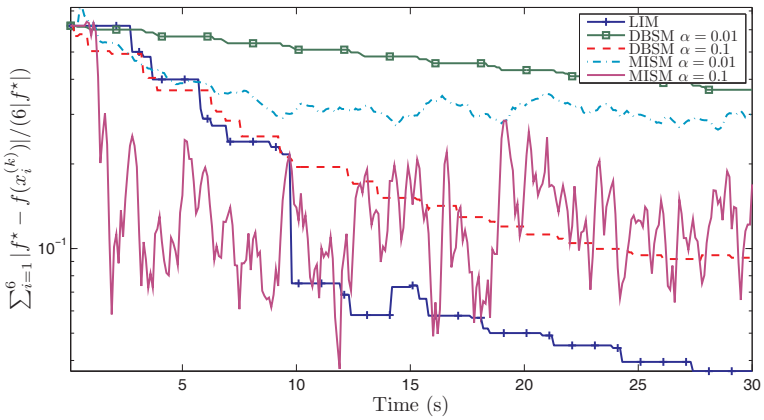


Figure 1.6: The plot shows the convergence (the curves should go to zero as fast as possible) versus number of iterations for different optimization algorithms implemented on a 6 mote WSN. The algorithms have inherently different communication patterns, which, in combination with the numerics of the algorithm, influence the performance. As can be seen, the convergence behavior depends quite a bit on the algorithm. This example will be discussed in detail in Chapter 6.

more, we use the following classification (inspired by Skogestad and Postlethwaite (2005, page 386)):

Open-loop optimization The system is controlled in open-loop and the inputs are chosen by solving an optimization problem off-line. This approach is very sensitive to modeling errors and disturbances. The setup is shown in Fig. 1.7a.

Set-point provided by optimization The system is controlled in closed-loop by a controller. Typically, the job of the controller is to keep the output of the system at a given set-point or to follow a given trajectory. The optimal set-point or trajectory is computed by solving an optimization problem off-line. Then the set-point or trajectory is followed by the controller. See Fig. 1.7b for a block diagram illustration.

Optimizing controller The system is directly controlled by solving an optimization problem on-line. This approach often requires a better model of the plant, compared with the model needed for synthesizing simpler controllers. The block diagram is shown in Fig. 1.7c.

Note that there are of course always cases that do not fit into this classification, and it is also possible to combine the previously mentioned classes.

In Chapter 4, we consider resource allocation in communication networks. The key idea in this chapter is to "view the TCP/IP network as an optimization solver,"

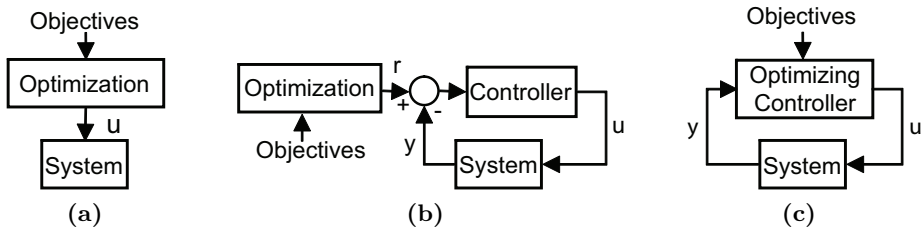


Figure 1.7: Different combinations of optimization and control. (a) The control signal is found by optimization and is applied in open-loop. (b) The set-point is provided by solving an optimization problem. The set-point is then achieved by a controller. (c) The control signal is found by solving an optimization problem on-line using feedback information.

using the words of Chiang et al. (2007). The optimal resource allocation in the network is posed as a static optimization problem, and if everything was known a priori, we could in principle solve the problem off-line and use approach Fig. 1.7a (adjust the resource allocation according to the optimal solution) or Fig. 1.7b (use the optimal solution as a set-point, which is achieved by a controller). However, everything about the network is not known, and it is impossible, or at least practically difficult, to collect all information. Therefore, we need to solve the optimization problem on-line, where the iterates are actually used in the network to evaluate the currently achieved utility. This approach corresponds to Fig. 1.7c.

In Chapter 5, we consider a multi-agent optimal rendezvous problem, where the rendezvous point is chosen (in fact, found through decentralized negotiations between the agents) such that the agents' trajectories minimize a quadratic cost. The basic setup is that the rendezvous point is found before any action is taken, which corresponds to Fig. 1.7b. However, the setup could potentially be run in the mode of Fig. 1.7c. In addition, we also consider the consensus problem where linear iterations should converge to the barycenter of the initial states. This scheme is optimized by choosing the coefficients of the linear iteration in an optimal way before the iteration is executed. This corresponds to off-line optimization of the algorithm itself, which is not exactly covered by Fig. 1.7.

In Chapter 6, we devise a generic optimization software component for WSNs, and we evaluate how some of the most promising optimization algorithms will actually perform in simulated as well as real networked systems. This means that Fig. 1.7 is not directly applicable, since we do not consider a specific application.

1.4 Outline and Contributions

We now describe the outline and contributions of the thesis in more detail. The applications and the corresponding related work are further and more thoroughly presented in each chapter.

1.4.1 Chapter 2

In this chapter, we present background material, such as fundamental definitions and algorithms, on which the rest of the thesis is built. In particular, we present basic convexity notions, some standard optimization algorithms, and decomposition techniques.

1.4.2 Chapter 3

Three novel optimization algorithms are presented in this chapter. More specifically, we consider an optimization problem were a convex and potentially non-smooth additive objective function should be minimized subject to convex constraints. The objective function consists of n terms, and each term is a function of the vector x. Furthermore, an n-node connected graph is associated with the optimization problem, and each node is associated with a term in the objective function. Therefore, the optimization problem can interpreted as a networked system; the nodes should cooperate to find the network-wide decision variable x. First, we develop an extension of the random incremental subgradient method due to Nedić and Bertsekas. Second, we develop a novel algorithm which is based on a combination of a subgradient algorithm interleaved with a fixed number of consensus iterations. Third, we consider a special case of the previously presented optimization problem; in this case, the objective function is assumed to be separable, as well, and we have a network-wide constant sum constraint. We propose an optimization algorithm for this problem class by extending the center-free resource allocation algorithm, due to Ho et al, to the non-smooth case.

The chapter is based on the following publications.

B. Johansson, M. Rabi, and M. Johansson. A simple peer-to-peer algorithm for distributed optimization in sensor networks. In *Proceedings of IEEE CDC* (2007)

B. Johansson, M. Rabi, and M. Johansson. A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM J. Optim.* (2008*d*). Submitted

B. Johansson, T. Keviczky, K. H. Johansson, and M. Johansson. Subgradient methods and consensus algorithms for solving convex optimization problems. In *Proceedings of IEEE CDC* (2008*b*)

1.4.3 Chapter 4

This chapter considers how to use decomposition techniques in conjunction with algorithms from Chapter 3 to perform cross-layer optimization in communication systems. We also present a flowchart that can be useful to categorize many of the existing approaches to cross-layer optimization existing in the literature. Furthermore, we devise two algorithms for cross-layer optimization in a frequency-division

multiple access and a spatial reuse time-division multiple access network, respectively.

The following publications provide the cornerstones for this chapter.

B. Johansson and M. Johansson. Primal and dual approaches to distributed cross-layer optimization. In *16th IFAC World Congress* (2005)

P. Soldati, B. Johansson, and M. Johansson. Proportionally fair allocation of end-to-end bandwidth in STDMA wireless networks. In *Proceedings of ACM MobiHoc* (2006*b*)

B. Johansson, P. Soldati, and M. Johansson. Mathematical decomposition techniques for distributed cross-layer optimization of data networks. *IEEE Journal on Selected Areas in Communications*, 24(8): 1535–1547 (2006*b*)

P. Soldati, B. Johansson, and M. Johansson. Distributed optimization of end-to-end rates and radio resources in wimax single-carrier networks. In *Proceedings of IEEE GLOBECOM* (2006*a*)

P. Soldati, B. Johansson, and M. Johansson. Distributed cross-layer coordination of congestion control and resource allocation in s-TDMA wireless networks. *Wireless Networks*, 14: 949–965 (2008)

B. Johansson, H. Li, J. Huang, M. Chiang, and M. Johansson. Network utility maximization website (2008*c*). URL `http://www.networkutilitymaximization.org/`

1.4.4 Chapter 5

In this chapter, we first consider a multi-agent rendezvous problem in which an optimal consensus point is found through decentralized negotiations between the agents in the system. The consensus point is optimal in the sense that the agents' trajectories to this point minimize a quadratic cost criteria. Second, we look into the canonical problem of consensus, i.e., how should linear iterations be devised such that a state vector at each node asymptotically converges to the barycenter of the nodes' initial state vectors. In particular, we investigate the potential benefit of an augmented state vector. Finally, we provide necessary and sufficient convergence conditions for a general augmented state vector linear iteration.

The chapter is founded on the publications below.

B. Johansson, A. Speranzon, M. Johansson, and K. H. Johansson. Distributed model predictive consensus. In *Mathematical Theory of Networks and Systems* (2006*c*)

B. Johansson, A. Speranzon, M. Johansson, and K. H. Johansson. On decentralized negotiation of optimal consensus. *Automatica*, 44: 1175–1179 (2008*e*)

B. Johansson and M. Johansson. Faster linear iterations for distributed averag-

ing. In *IFAC World Congress* (2008)

1.4.5 Chapter 6

In this chapter, we show how to implement some of the presented algorithms on a real networked system, namely a WSN. First, we implement the algorithms in the network simulator NS2 and perform detailed Monte Carlo simulations. Second, we implement the algorithms on the Tmote Sky mote, a specific wireless sensor, and evaluate the performance in experiments.

The chapter is based on the following publications.

B. Johansson, C. M. Carretti, and M. Johansson. On distributed optimization using peer-to-peer communications in wireless sensor networks. In *Proceedings of IEEE SECON* (2008*a*)

The NS2 simulations and the major part of the WSN coding were made by Cesare Maria Carretti. The NS2 simulations are described in detail in the following master thesis, which was supervised by the author.

C. M. Carretti. *Comparison of Distributed Optimization Algorithms in Sensor Networks*. Master's thesis, Royal Institute of Technology (KTH) (2008)

1.4.6 Chapter 7

In this chapter, we summarize the thesis and discuss the results. We also outline the natural steps to continue the work started with this thesis.

1.4.7 Other Publications

The following publications are not explicitly part of this thesis, but they have influenced the contents.

A. Speranzon, C. Fischione, B. Johansson, and K. H. Johansson. Adaptive distributed estimation over wireless sensor networks with packet losses. In *Proceedings of IEEE CDC* (2007)

B. Johansson, C. Adam, M. Johansson, and R. Stadler. Distributed resource allocation strategies for achieving quality of service. In *Proceedings of IEEE CDC*, 1990–1995 (2006*a*)

B. Johansson and M. Gäfvert. Untripped SUV rollover detection and prevention. In *Proceedings of IEEE CDC* (2004)

Chapter 2

Preliminaries

"*Convexity thus plays a role much the same as that of linearity in the study of dynamical systems.*"

L. S. Lasdon, Optimization Theory for Large Systems, 1970.

In this chapter, we present the mathematical building blocks, on which this thesis is built. The presentation is at times rather brief, but there are pointers to the relevant literature. The detailed outline of the chapter is as follows. In Section 2.1, we introduce important notions for convex optimization. Then, in Section 2.2, we present some graph related notation that we will use. Section 2.3 introduces so-called peer-to-peer (nearest neighbor) algorithms and consensus algorithms. In Section 2.4, we review some theory regarding Markov chains. Then, in Section 2.5, a generic resource allocation problem is introduced, as well as its optimality conditions. Section 2.6 reviews related and relevant standard optimization methods. In Section 2.7, we describe some decomposition techniques that can be used to decompose an optimization problem. Finally, we summarize the chapter in Section 2.8.

2.1 Convex Optimization

This section introduces important convexity notions; see, e.g., Boyd and Vandenberghe (2004) or the classic Rockafellar (1970), for a more thorough exposition.

Definition 2.1.1. *A set, $\mathcal{X} \subseteq \mathbb{R}^\xi$, is convex if*

$$(\alpha x + (1-\alpha)y) \in \mathcal{X}$$

for all x and y in $\mathcal{X}$ and $0 \leq \alpha \leq 1$.

Definition 2.1.2. *A function, $f : \mathcal{X} \subseteq \mathbb{R}^\xi \to \mathbb{R}$, where the domain $\mathcal{X}$ is a convex set, is convex if*

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$$

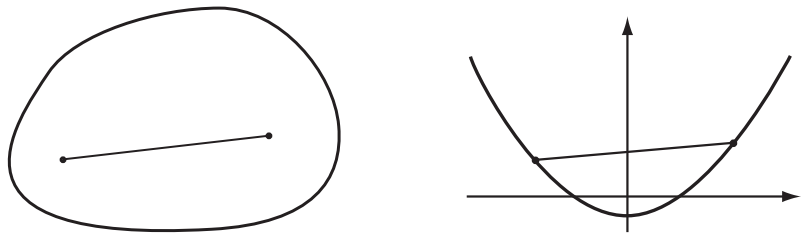


Figure 2.1: a) Example of a convex set. b) Example of a convex function.

for all x and y in $\mathcal{X}$ and $0 \leq \alpha \leq 1$. If $-f$ is convex, then f is concave.

A convex set is illustrated in Fig. 2.1a and a convex function is illustrated in Fig. 2.1b.

If the objective function in a minimization problem is convex and the feasible set is convex, then we have a *convex optimization problem*. Likewise, maximization of a concave function over a convex set is also a convex optimization problem, since $\inf_{x \in \mathcal{X}} f(x) = \sup_{x \in \mathcal{X}} -f(x)$.

The convexity assumption is restrictive, but convex optimization problems have a very attractive property: a local solution is also a global solution. We have the following lemma (see, e.g., Bertsekas et al. (2003, Proposition 2.1.2)).

Lemma 2.1.3. *A local minimum for a convex optimization problem is also a global minimum for the same problem.*

Convex optimization problems also have some useful optimality conditions; see Section 2.7.1. Furthermore, surprisingly many relevant engineering problems can be posed as, or at least approximated with, convex optimization problems. Numerous examples are provided in Boyd and Vandenberghe (2004).

Not all objective functions are differentiable and to handle this case, we make the following definition.

Definition 2.1.4. *A vector $a \in \mathbb{R}^\xi$ is a subgradient of a convex function $f : \mathbb{R}^\xi \to \mathbb{R}$ at a point $x \in \mathbb{R}^\xi$ if*

$$f(y) \geq f(x) + a^\intercal (y - x), \quad \forall y \in \mathbb{R}^\xi, \tag{2.1}$$

where $^\intercal$ denotes the transpose operator. The set of all subgradients of a convex function f at $x \in \mathbb{R}^\xi$ is called the subdifferential of f at x, and is denoted by $\partial f(x)$:

$$\partial f(x) = \left\{ a \in \mathbb{R}^\xi | f(y) \geq f(x) + a^\intercal (y - x), \forall y \in \mathbb{R}^\xi \right\}. \tag{2.2}$$

Note that it follows from the definition that a subgradient can be used to form an affine global underestimator of the convex function. Furthermore, if the subdifferential at x only contains one element, the function f is differentiable at x with $\nabla f(x) = \partial f(x)$ (Bertsekas et al., 2003, Proposition 4.2.2).

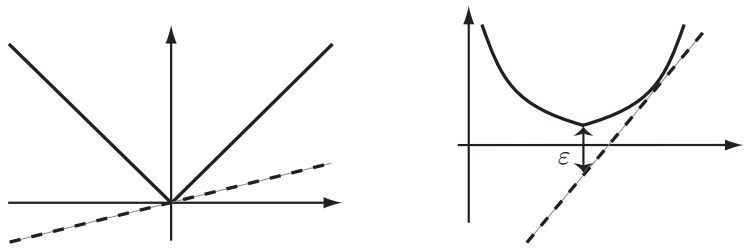


Figure 2.2: a) The solid line is $y = |x|$ and the dashed line, $y = \frac{x}{4}$, is drawn in the direction of one of the corresponding subgradients $\frac{1}{4}$. b) Example of an ε-subgradient.

Example 2.1.1. A simple example of a convex non-differentiable function is $f(x) = |x|, x \in \mathbb{R}$. This function is non-differentiable at $x = 0$ and $\partial f(0) = \{a \in \mathbb{R} | -1 \leq a \leq 1\}$; see Fig. 2.2a for an illustration.

Definition 2.1.5. *Let the scalar $\varepsilon \geq 0$. A vector $a \in \mathbb{R}^\xi$ is an ε-subgradient of a convex function $f : \mathbb{R}^\xi \to \mathbb{R}$ at a point $x \in \mathbb{R}^\xi$ if*

$$f(y) \geq f(x) + a^\intercal (y - x) - \varepsilon, \quad \forall y \in \mathbb{R}^\xi. \tag{2.3}$$

The ε-subdifferential of a convex function f at $x \in \mathbb{R}^\xi$ is the set of ε-subgradients:

$$\partial_\varepsilon f(x) = \left\{ a \in \mathbb{R}^\xi | f(y) \geq f(x) + a^\intercal (y - x) - \varepsilon, \forall y \in \mathbb{R}^\xi \right\}. \tag{2.4}$$

An example of an ε-subgradient is shown in Fig. 2.2b. We also have the following additional notions: the conditional subdifferential and the conditional ε-subdifferential, which were first introduced by Dem'yanov and Shomesova (1980).

Definition 2.1.6. *Let the scalar $\varepsilon \geq 0$ and the function $f : \mathcal{X} \to \mathbb{R}$ be convex, where $\mathcal{X} \subseteq \mathbb{R}^\xi$ is a closed and convex set. The conditional subdifferential at x, $\partial^{\mathcal{X}} f(x)$, is defined as*

$$\partial^{\mathcal{X}} f(x) = \left\{ a \in \mathbb{R}^\xi | f(y) \geq f(x) + a^\intercal (y - x) - \varepsilon, \forall y \in \mathcal{X} \right\}. \tag{2.5}$$

The conditional ε-subdifferential at x, $\partial^{\mathcal{X}}_\varepsilon f(x)$, is defined as

$$\partial^{\mathcal{X}}_\varepsilon f(x) = \left\{ a \in \mathbb{R}^\xi | f(y) \geq f(x) + a^\intercal (y - x) - \varepsilon, \forall y \in \mathcal{X} \right\}. \tag{2.6}$$

Using the conditional subdifferential, some optimality conditions for constrained minimization becomes quite elegant (Dem'yanov and Shomesova, 1980, Lemma 4):

$$x^\star \in \left\{ x \in \mathcal{X} \middle| f(x) = \inf_{x \in \mathcal{X}} f(x) \right\} \Leftrightarrow 0 \in \partial^{\mathcal{X}} f(x^\star).$$

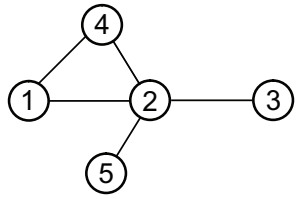


Figure 2.3: An example graph with 5 nodes (vertices) and 5 links (edges).

2.2 Graphs

For our developments, we will use some graph notation and a very small subset of the rich spectrum of available graph theory results. More details on graphs can be found in, e.g., Diestel (2005) and Godsil and Royle (2001).

A *graph*, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, consists of a *vertex* set, $\mathcal{V}$, and an *edge* set, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. An edge is an unordered pair of distinct vertices of $\mathcal{G}$. We consider graphs only with finitely many vertices. A graph is *connected* if any two vertices in the graph have a path between them. The *adjacency matrix* $A(\mathcal{G})$ of a graph $\mathcal{G}$ is a symmetric 01-matrix. Furthermore, the elements are defined as $[A(\mathcal{G})]_{ij} = 1$ if $(i,j) \in \mathcal{E}$ and $[A(\mathcal{G})]_{ij} = 0$ otherwise. The set of neighbors of a vertex i in $\mathcal{G}$ is denoted $\mathcal{N}(i) = \{j \in \mathcal{V} \,|\, (i,j) \in \mathcal{E}\}$. The number of neighbors of a vertex (or node) is called the degree, $d_i(\mathcal{G}) = |\mathcal{N}(i)|$ or d_i if the underlying graph is understood by the context. The *Laplacian* matrix $L(\mathcal{G})$ of a graph $\mathcal{G}$ is defined as $L(\mathcal{G}) = D(\mathcal{G}) - A(\mathcal{G})$, where $D(\mathcal{G})$ is a diagonal matrix with $[D(\mathcal{G})]_{ii} = d_i(\mathcal{G})$. Furthermore, we denote the maximum degree $\Delta(\mathcal{G}) = \max_{i \in \mathcal{V}} d_i(\mathcal{G})$. The adjacency matrix and the Laplacian matrix have some interesting properties from which you can say a great deal about the graph (see, e.g., Godsil and Royle (2001)), but we will not go into those details here. Finally, it is also possible to use *directed* graphs, where the edges have an orientation, but we will not use such graphs in this thesis.

Example 2.2.1. An example graph, $\mathcal{G}$, is shown Fig. 2.3. It has the following adjacency matrix and Laplacian matrix,

$$A(\mathcal{G}) = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \text{ and } L(\mathcal{G}) = \begin{pmatrix} 2 & -1 & 0 & -1 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ 0 & -1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 2 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{pmatrix}.$$

2.3 Peer-to-Peer Algorithms and Consensus

The optimization problems we consider in this thesis have a corresponding undirected connected graph, $\mathcal{G}$, where each node (also called agent or vertex) is associated with some part of the optimization problem, e.g., a term in the objective function or an element in the decision variable. Furthermore, the nodes in the graph can only communicate with neighboring nodes, and the edges can thus be interpreted as bidirectional communication links. More formally, the interpretation is that if $(i, j) \in \mathcal{E}$, then node i can communicate with node j. We consider iterative algorithms that solve such optimization problems by letting the nodes cooperate with their neighbors; we denote such algorithms to be *peer-to-peer* algorithms. All algorithms that we devise and analyze in this thesis belong to this class of algorithms. Since we would like to find a globally optimal solution to the optimization problem, all nodes need to coordinate with all other nodes, at least implicitly[1]. In view of this, we make the following rather natural assumption.

Assumption 2.3.1. *The graph, $\mathcal{G}$, is connected.*

Our notion of a peer-to-peer algorithm fits into the broad framework for distributed computation that was first proposed in Tsitsiklis et al. (1986), and it has been used by many others since then. Tsitsiklis et al. (1986) consider a rather general communication case, namely asynchronous communications; we on the other hand, as we will see later, address the more specific synchronous communication case. In addition, they focus on gradient algorithms, whereas we will consider the more general case of subgradient algorithms; see Chapter 3.

The *consensus problem* considers conditions under which using a certain message-passing protocol, the local variables of each agent will converge to the same value. However, this value does not generally represent an optimizer of a problem of interest, and is typically a function (e.g., the average) of the initial values held by the agents and the information exchange policy. The field was pioneered by DeGroot (1974), and nowadays, a wide variety of results exist related to the convergence of local variables to a common value using various information exchange procedures among multiple agents; see, e.g., the informative survey by Olfati-Saber et al. (2007) and the references therein.

For our developments, we will consider the following consensus iteration in discrete-time:

$$y^{(k+1)} = Wy^{(k)}, \tag{2.7}$$

where the ith element of vector $y^{(k)} \in \mathbb{R}^n$, denoted by $[y^{(k)}]_i$, corresponds to agent i. The information exchange among agents is represented by the matrix W, for which we make the following assumptions.

[1]Explicit coordination between all nodes is of course possible, but it will require that the network is fully connected (every node can communicate directly with all nodes) or that message relaying is provided in some way.

Assumption 2.3.2 (Consensus Matrix Properties)**.** *The matrix $W \in \mathbb{R}^{n\times n}$, with $n = |\mathcal{V}|$, fulfills*

$$\begin{aligned} &a)\ [W]_{ij} = 0,\ \text{if } (i,j) \notin \mathcal{E} \text{ and } i \neq j, \\ &b)\ \mathbf{1}_n^\intercal W = \mathbf{1}_n^\intercal, \quad c)\ W\mathbf{1}_n = \mathbf{1}_n, \quad d)\ \rho\left(W - \mathbf{1}_n\mathbf{1}_n^\intercal/n\right) \leq \gamma < 1, \end{aligned}$$

where $\mathbf{1}_n \in \mathbb{R}^n$ is the column vector with all elements equal to one and $\rho(\cdot)$ is the spectral radius.

The sparsity constraints on W can be summarized as $W \in \mathcal{W}$, where

$$\mathcal{W} \triangleq \{W \in \mathbb{R}^{n\times n} | [W]_{ij} = 0,\ \text{if } (i,j) \notin \mathcal{E} \text{ and } i \neq j\}. \tag{2.8}$$

It turns out that these assumptions are necessary and sufficient for consensus; see, e.g., Xiao and Boyd (2004, Theorem 1).

Lemma 2.3.3. *If and only if Assumption 2.3.2 b)-d) is fulfilled, then*

$$\lim_{k\to\infty} W^k = \frac{\mathbf{1}_n\mathbf{1}_n^\intercal}{n} \text{ and } \lim_{k\to\infty} y^{(k)} = \bar{y}^{(0)}.$$

The matrix W can for example be chosen as $W = I - \varepsilon L(\mathcal{G})$, which is the so-called Perron matrix of the associated graph $\mathcal{G}$ with parameter ε. We have the following result, which is an existence result for W fulfilling Assumption 2.3.2; see, e.g., Olfati-Saber et al. (2007, Theorem 2).

Lemma 2.3.4. *If the graph, $\mathcal{G}$, is connected and if $W = I - \varepsilon L(\mathcal{G})$ with $0 < \varepsilon < 1/\Delta(\mathcal{G})$, then W fulfills Assumption 2.3.2 b)-d) and Lemma 2.3.3 applies.*

The drawback with the Perron matrix is that it requires global information, namely the value of ε. However, there is a rather simple and distributed scheme that will give us a W fulfilling Assumption 2.3.2, based on the so-called Metropolis-Hastings scheme (Hastings, 1970); see Boyd et al. (2004).

Lemma 2.3.5 (Metropolis-Hastings)**.** *If the graph, $\mathcal{G}$, is connected, then W fulfills Assumption 2.3.2, if the elements of W are set to*

$$[W]_{ij} = \begin{cases} \min\{d_i^{-1}, d_j^{-1}\} & \text{if } (i,j) \in \mathcal{E} \text{ and } i \neq j \\ \sum_{(i,k)\in\mathcal{E}} \max\{0, d_i^{-1} - d_k^{-1}\} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} \tag{2.9}$$

The convergence behavior of W^k to $\mathbf{1}_n\mathbf{1}_n^\intercal/n$ can vary quite a bit depending on how W is chosen, as well as on the underlying graph; there exist different metrics,

which can be used to quantify the speed of convergence. The most commonly used metrics are the per-step convergence factor,

$$\sup_{x\neq 0} \frac{\|(W - \mathbf{1}_n\mathbf{1}_n^\intercal/n)x\|}{\|x\|} = \|(W - \mathbf{1}_n\mathbf{1}_n^\intercal/n)x\|\,,$$

which is the definition of the spectral norm, and the worst case geometric average convergence factor (Varga, 1962, Theorem 3.2),

$$\lim_{k\to\infty}\left(\sup_{x\neq 0} \frac{\left\|(W - \mathbf{1}_n\mathbf{1}_n^\intercal/n)^k x\right\|}{\|x\|}\right)^{1/k} = \lim_{k\to\infty} \left\|(W - \mathbf{1}_n\mathbf{1}_n^\intercal/n)^k\right\|^{1/k} = \rho(W - \mathbf{1}_n\mathbf{1}_n^\intercal/n).$$

In addition, we have that $\rho(A) \leq |||A|||$, for any square matrix A and any matrix norm $|||\cdot|||$, including the spectral norm $\|\cdot\|$ (Horn and Johnson, 1985, Theorem 5.6.9). Clearly, the worst case per step convergence factor is always worse than the worst case geometric average convergence factor. If we restrict ourselves to symmetric W, the spectral norm and the spectral radius coincide. Furthermore, it is possible to efficiently find the optimal symmetric matrix, in the sense that $\|W - \mathbf{1}_n\mathbf{1}_n^\intercal/n\|$ is minimal, using the following semi-definite program (SDP) (Xiao and Boyd, 2004),

$$\begin{array}{ll} \underset{W,s}{\text{minimize}} & s \\ \text{subject to} & -sI \leq W - \mathbf{1}_n\mathbf{1}_n^\intercal/n \leq sI \\ & [W]_{ij} = 0, \text{ if } (i,j) \notin \mathcal{E} \text{ and } i \neq j \\ & W\mathbf{1}_n = \mathbf{1}_n, \quad W^\intercal = W, \end{array} \tag{2.10}$$

where the inequality $-sI \leq W - \mathbf{1}_n\mathbf{1}_n^\intercal/n$, a so-called linear matrix inequality (LMI), indicates that the matrix $W - \mathbf{1}_n\mathbf{1}_n^\intercal/n + sI$ is positive semi-definite, and so forth. This optimization problem can easily be set up in Matlab using the software package Yalmip (Löfberg, 2004) and efficiently solved with an SDP solver, such as SeDuMi (SeDuMi, 2008).

2.4 Markov Chains and Fast Mixing

Markov chains are stochastic processes without memory: only the current state influence the future path. More formally, a sequence of random variables $\{x_k\}_{k=0}^{\infty}$ with values in the set $\{1, ..., n\}$ is a *Markov chain*, MC, if

$$\mathbb{P}\{x_{k+1} = j | x_k = i,\, x_{k-1} = i_{k-1},\, ...,\, x_0 = i_0\} = \mathbb{P}\{x_{k+1} = j | x_k = i\},$$

for all non-negative integers k and all states $i_0, ..., i_{k-1}, i, j$, where $\mathbb{P}\{\cdot\}$ is the probability of the event $\{\cdot\}$. If the following holds for all positive integers k

$$\mathbb{P}\{x_{k+1} = j | x_k = i\} = \mathbb{P}\{x_k = j | x_{k-1} = i\},$$

we have a *homogeneous* Markov chain, HMC. For a more detailed exposition, see, e.g., Norris (1998) or Brémaud (1999). Many useful properties, characterized using matrices, of finite Markov chains are given in Kemeny and Snell (1960). We will now go through some more notions and properties that will be very useful later. The transition matrix P of an HMC is defined as $[P]_{ij} = \mathbb{P}\{x_{k+1} = j | x_k = i\}$, and it fulfills $\sum_{i=j}^{n} [P]_{ij} = 1$ and $[P]_{ij} \geq 0$. An MC is *irreducible* if $[P^k]_{ij} > 0$ for some $k \geq 1$ for all $i, j \in \mathcal{V}$. Furthermore, an MC is *aperiodic* if $[P^k]_{ii} > 0$ for sufficiently large $k \geq 1$ for all $i \in \mathcal{V}$.

An integer random variable, τ, for which the event $\{\tau = l\}$ only depends on the random variables $\{x_k\}_{k=0}^{l}$, is called a *stopping time* with respect to $\{x_k\}_{k=0}^{\infty}$. More specifically, we have $I\{\tau = l\} = \psi(x_0, x_1, ..., x_l)$, where $I\{\cdot\}$ is the indicator function of the event $\{\cdot\}$. We have the following useful lemma; see, e.g., Theorem 7.1 in Brémaud (1999).

Lemma 2.4.1 (Strong Markov property)**.** *Let τ be a stopping time with respect to the HMC $\{x_k\}_{k=0}^{\infty}$ with transition matrix P. Then, the process after τ and the process before τ are independent. Furthermore, the process after τ is an HMC with transition matrix P.*

If the probability of jumping between two states is zero whenever the corresponding nodes in the graph are not connected, we can use this MC to model a stochastic peer-to-peer algorithm. For precisely this reason, as we will see in Section 3.3, it is useful to be able to construct an MC with uniform stationary distribution, where each state corresponds to nodes in a graph. Furthermore, the properties of P are closely related to the properties of W previously presented in Section 2.3. The only difference is that we have the extra condition that the elements in the transition matrix should be non-negative. This enables us to use the powerful theory of Perron-Frobenius (Horn and Johnson, 1985). For our purposes, the transition matrix P should fulfill the following.

Assumption 2.4.2 (Transition Matrix Properties)**.** *The matrix $P \in \mathbb{R}^{n \times n}$, with $n = |\mathcal{V}|$, fulfills*

a) $[P]_{ij} = 0$, *if* $(i, j) \notin \mathcal{E}$ *and* $i \neq j$,
b) $[P]_{ij} \geq 0$, $\forall i, j$
c) $\mathbf{1}_n^\intercal P = \mathbf{1}_n^\intercal$, *d)* $P\mathbf{1}_n = \mathbf{1}_n$, *e)* $\rho\left(P - \mathbf{1}_n\mathbf{1}_n^\intercal/n\right) \leq \gamma < 1$.

We have the following lemma.

Lemma 2.4.3. *If Assumption 2.4.2 b)-e) is fulfilled, then P is a transition matrix with*

$$\lim_{k \to \infty} P^k = \frac{\mathbf{1}_n\mathbf{1}_n^\intercal}{n}.$$

We assume that the graph $\mathcal{G}$ is connected in this section as well, and this implies that there is a (multihop) path between any pair of nodes. How do we to construct, using only local information, the transition matrix of MC, P, such that the iterate, x_k, only jumps to an adjacent and reachable node? Since the Perron matrix and the Metropolis-Hastings scheme presented in Section 2.3 yield a weight matrix W with non-negative elements, we get a P that fulfills 2.4.2 if we set $P = W$.

It is also possible to devise an optimal symmetric transition matrix using the following semi-definite program (SDP),

$$\begin{array}{ll} \underset{P,s}{\text{minimize}} & s \\ \text{subject to} & -sI \leq P - \mathbf{1}_n\mathbf{1}_n^\intercal/n \leq sI \\ & [P]_{ij} = 0, \text{ if } (i,j) \notin \mathcal{E} \text{ and } i \neq j \\ & [P]_{ij} \geq 0, \forall i,j = 1,...,n, \quad P\mathbf{1}_n = \mathbf{1}_n, \quad P^\intercal = W, \end{array} \tag{2.11}$$

which is the same as (2.10) except an additional non-negativity constraint on each element in P. This optimal matrix will be the fastest converging (sometimes called fastest *mixing*) symmetric matrix having a uniform stationary distribution and fulfilling the sparsity constraints.

2.5 Centralized Optimal Resource Allocation

In this section, we review some properties of a simple resource allocation problem that will turn out to be useful later on in the thesis. First, we consider the case when the allocated resources are continuous variables. Second, we add the constraint that the allocated resources have to be integers.

2.5.1 Convex Case

The simple resource allocation problem

$$\begin{array}{ll} \underset{x}{\text{maximize}} & \sum_{i=1}^n u_i(x_i) \\ \text{subject to} & \sum_{i=1}^n x_i = x_{\text{tot}} \\ & x_i \geq 0, \ i = 1,...,n, \end{array} \tag{2.12}$$

is a special instance of the more general problem (2.27), which we will present later. Furthermore, we assume $x_i \in \mathbb{R}$ for all $i = 1,...,n$. The problem can be interpreted as that all resources should be used and they should be allocated to maximize the sum of the utilities for each resource. This is a standard problem in economics and it is central in order to solve some of the subproblems we will encounter later. Several solution algorithms exist when $u_i(x_i)$ is strictly concave and twice differentiable.

The optimal point of (2.12) can be characterized by the Karush-Kuhn-Tucker conditions (see (2.33e) or, e.g., Boyd and Vandenberghe (2004))[2]

$$\begin{cases} u_i'(x_i^\star) = \psi^\star \text{ if } x_i^\star > 0 \\ u_i'(x_i^\star) \leq \psi^\star \text{ if } x_i^\star = 0 \\ \sum_i x_i^\star = x_{\text{tot}}. \end{cases} \tag{2.13}$$

This means that at the optimal point, the slopes of the utility functions will be the same, except for those resources that are equal to zero. Since the utility functions are concave, their slopes are decreasing, and if one utility function has a higher slope than the other utility functions for some allocation, then that utility function should get more resources.

As we will see in Chapter 4, these properties can be exploited in the distributed search for the optimal point.

2.5.2 Discrete Case

In some applications where the resources are inherently discrete, we have to add the constraint that the x variables in (2.12) have to be integers. We then get the following problem

$$\begin{array}{ll} \underset{x}{\text{maximize}} & \sum_{i=1}^n u_i(x_i) \\ \text{subject to} & \sum_{i=1}^n x_i = x_{\text{tot}} \\ & x_i \in \{1, 2, 3, ..., x_{\text{tot}}\},\ i = 1, ..., n. \end{array} \tag{2.14}$$

This problem is nonconvex, which in general implies that it is hard to verify if a local optimal solution is globally optimal. However, in this case, with $u_i(\cdot)$ concave or strictly concave, it is in fact possible to quite easily find the globally optimal allocation. This case is a so-called discrete resource allocation problem with a separable and concave objective function.

To describe the optimal allocation, we define[3] $\Delta_i(x_i)$, the marginal utility, for integer x_i greater than zero, as

$$\Delta_i(x_i) = u_i(x_i) - u_i(x_i - 1).$$

Thus, $\Delta_i(x_i)$ is the increase in utility when one resource is added to $x_i - 1$ to reach x_i, and $u_i(x_i) = u_i(0) + \sum_{i=1}^{x_i} \Delta_i(i)$, $x_i \in \{1, ..., x_{\text{tot}}\}$. Furthermore, since $u_i(\cdot)$ is concave or strictly concave, we have that

$$\Delta_i(1) \geq \Delta_i(2) \geq \cdots \geq \Delta_i(x_{\text{tot}}). \tag{2.15}$$

[2] According to Danskin (1967), the conditions for this problem are also known as Gibbs lemma, after J. Willard Gibbs, who laid the foundation of modern chemical thermodynamics in *Equilibrium of Heterogenous substances*, 1876. However, according to Patriksson (2008), the theorem was first proved using diagrams by Hermann H. Gossen in *Die Entwicklung der Gesetze des menschlichen Verkehrs und der daraus fliessenden Regeln für menschliches Handeln*, 1854.

[3] This definition is not related to $\Delta(\mathcal{G})$.

The optimal allocation consists of the x_{tot} largest elements in the set of all possible Δ's (see Ibaraki and Katoh (1988, Theorem 4.1.1)). Since the Δ's are sorted for each x_i by (2.15), we can find the largest elements in a greedy fashion. The greedy algorithm, which is one of the most well-known algorithms to find the optimal allocation, is now as follows: starting from a zero allocation, the algorithm adds a resource one at a time. One resource is added to the variable that have the greatest marginal utility. The algorithm stops when the total number of resources are allocated. More efficient algorithms also exist, but they are more complicated; see, e.g., Ibaraki and Katoh (1988) for details.

If we define

$$\Delta_i(0) = \infty,$$

then we can express the optimal allocation of (2.14) in a similar way to (2.13),

$$\begin{cases} \Delta_i(x_i^\star) \geq \psi & \text{for all } i \\ \Delta_i(x_i^\star + 1) \leq \psi & \text{for all } i \\ \sum_i x_i^\star = x_{\text{tot}} & \\ x_i^\star \in \{0, 1, ..., x_{\text{tot}}\} & \text{for all } i. \end{cases} \tag{2.16}$$

This is a direct consequence of that the optimal allocation consists of the x_{tot} largest elements of the Δ's. The ψ variable is a threshold between the x_{tot} largest elements and the other smaller elements. This means that an interval can be optimal, i.e., $\psi^\star \in \Psi^\star$, whereas in the convex case, the threshold, $\psi^\star$, is a fixed number.

2.6 Optimization Algorithms

A huge number of optimization algorithms have been devised through the years, and we will not make any attempt at all to provide a survey here. We will, however, go through some of the most standard algorithms for general convex optimization problems that are relevant for this thesis, as well as more specialized distributed (also relevant) algorithms, which exploit extra information about the structure of the problem. We would like to emphasize that without assumptions (such as convexity) on the objective function and the feasible set, any attempt to construct a efficient general solution algorithm seems to be rather futile. This is due to the so-called No Free Lunch Theorem of Optimization, which is an impossibility theorem that tells us that a general-purpose universal optimization strategy is impossible; see, e.g., the very interesting and readable discussion in Ho and Pepyne (2002). Finally, a more thorough background on the methods presented here are given in Boyd and Vandenberghe (2004), Bertsekas et al. (2003), or Bertsekas (1999).

Algorithm 1 Subgradient Method

1: Let $k \leftarrow 0$ and $x^{(0)} \in \mathcal{X}$.
2: **loop**
3: Compute a subgradient at $x^{(k)}$.
4: Compute $x^{(k+1)}$ via (2.18) and let $k \leftarrow k+1$.
5: **end loop**

2.6.1 Gradient and Subgradient Methods

In this section we consider

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & x \in \mathcal{X}, \end{aligned} \tag{2.17}$$

where the function $f : \mathbb{R}^\xi \to \mathbb{R}$ is convex and the set $\mathcal{X} \subseteq \mathbb{R}^\xi$ is convex and closed. One of the simplest methods for solving (2.17) is the gradient descent algorithm, which attempts to maximize the decrease of the objective function in each iteration by updating the current iterate in the opposite direction of the gradient of f. If a fixed stepsize is used, the new iterate may be outside of the feasible set $\mathcal{X}$; if that is the case, then the iterate is typically projected on the feasible set $\mathcal{X}$, and the new iterate is set to be the projection. In many cases, however, f is non-differentiable. A natural extension of gradients for non-differentiable functions are *subgradients*; see Definition 2.1.4.

Subgradients can be used instead of the gradient to find the minimum, and these subgradient methods take steps in the opposite direction of a subgradient, i.e., proceed according to

$$x^{(k+1)} = \mathcal{P}_{\mathcal{X}} \left[x^{(k)} - \alpha^{(k)} a^{(k)}(x^{(k)}) \right], \tag{2.18}$$

where $\mathcal{P}_{\mathcal{X}}[\cdot]$ denotes projection on the convex and closed set $\mathcal{X}$ and $a^{(k)}(x^{(k)}) \in \partial f(x^{(k)})$. If the projection on the feasible set $\mathcal{X}$ or the computation of the subgradient is expensive, then the execution of this algorithm will be very expensive, since it requires many projections and subgradients. The subgradient method is outlined in Algorithm 1.

Contrary to the gradient method, the objective value does not necessarily decrease in each iteration of the subgradient method. Rather, the distance between the iterate and the optimal solution will decrease. To show asymptotic convergence, it is enough to use diminishing stepsizes and that the subgradients are bounded. Typically, the stepsizes are chosen according to

$$\alpha^{(k)} \geq 0, \quad \sum_{k=1}^{\infty} \alpha^{(k)} = \infty, \quad \sum_{k=1}^{\infty} \left(\alpha^{(k)} \right)^2 < \infty, \tag{2.19}$$

and this condition is satisfied with, e.g., $\alpha^{(k)} = 1/k$. The following lemma formalizes the previous discussion on convergence using such stepsizes; see, e.g., Bertsekas et al. (2003, Proposition 8.2.6).

Lemma 2.6.1. *If a solution to problem* (2.17) *exists, the subgradients are bounded by some constant* φ*, and the stepsizes are chosen to fulfill* (2.19)*, then Algorithm 1 converges,* $\lim_{k\to\infty} x^{(k)} = x^\star$.

If a fixed stepsize is used, i.e., $\alpha^{(k)} = \alpha$, then the best value so far will converge to an area around the optimal point as pointed out in the following lemma; see, e.g., Bertsekas et al. (2003, Proposition 8.2.2).

Lemma 2.6.2. *If a solution to problem* (2.17) *exists, the subgradients are bounded by some constant* φ*, and the stepsizes are set to a constant,* $\alpha^{(k)} = \alpha$*, then the best value of Algorithm 1 converges to a ball around the optimal point,*

$$\liminf_{k\to\infty} f(x^{(k)}) \leq f(x^\star) + \frac{\alpha\varphi^2}{2}.$$

If the objective function is differentiable, then ordinary projected gradient methods can be used, which basically are identical with (2.18), except that the subgradient is replaced by the gradient. If the gradient also is Lipschitz, i.e., there exists $\vartheta \geq 0$ such that $||\nabla f(x) - \nabla f(y)|| \leq \vartheta ||x - y||, \forall x, y \in \mathcal{X}$, then a projected gradient method with fixed stepsize can achieve convergence to an optimizer. In this method, the subgradient is replaced by the gradient, as previously mentioned, and the diminishing stepsize is replaced with a fixed stepsize. Convergence of this method is shown in the following lemma; see, e.g., Bertsekas (1999).

Lemma 2.6.3. *If a solution to problem* (2.17) *exists, the gradient is Lipschitz with Lipschitz constant* ϑ*, and the stepsize,* α*, fulfills* $0 < \alpha \leq \frac{\vartheta}{2}$*, then Algorithm 1 converges,* $\lim_{k\to\infty} x^{(k)} = x^\star$.

If the Lipschitz property does not hold, then a search can be done to find a stepsize that decreases the function value in each iteration. However, this requires global coordination and does not appear to be a viable alternative to use in the applications we have in mind.

2.6.2 Conditional Gradient Method

Instead of using the projected gradient method, it is possible to use the so-called *conditional gradient method* (Bertsekas, 1999). We consider again the problem (2.17) with the additional requirement that the set $\mathcal{X}$ is also compact and that the function f is differentiable. The update equation of the conditional gradient method is

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)}(\hat{x}^{(k)} - x^{(k)}), \tag{2.20}$$

Algorithm 2 Conditional Gradient Method

1: Let $k \leftarrow 0$ and $x^{(0)} \in \mathcal{X}$.
2: **loop**
3: Compute $\hat{x}^{(k)}$ according to (2.21).
4: Compute $\alpha^{(k)}$ according to (2.22).
5: Let $x^{(k+1)} \leftarrow x^{(k)} + \alpha^{(k)}(\hat{x}^{(k)} - x^{(k)})$ and $k \leftarrow k + 1$.
6: **end loop**

where $\alpha^{(k)}$ is a stepsize and

$$\hat{x}^{(k)} = \arg\min_{x \in \mathcal{X}} \nabla f(x^{(k)})^{\intercal} x. \tag{2.21}$$

The stepsize $\alpha^{(k)}$ can be chosen in different ways, but a typical way is to use

$$\alpha^{(k)} = \arg\min_{\alpha \in [0,1]} f\left(x^{(k)} + \alpha^{(k)}(\hat{x}^{(k)} - x^{(k)})\right). \tag{2.22}$$

Dunn and Harshbarger (1978) consider the conditional gradient method in a Banach space setting, where $\alpha^{(k)}$ is chosen according to an open-loop scheme. The term open-loop refers to that the choice of $\alpha^{(k)}$ does not depend on local properties of f. This type of scheme will turn out to be useful in Chapter 4. For our purposes, $\mathcal{X} \subset \mathbb{R}^{\xi}$ is general enough, and we have the following result.

Lemma 2.6.4 (Adapted from Dunn and Harshbarger (1978, Theorem 1))**.** *Let $\mathcal{X} \subset \mathbb{R}^{\xi}$ be a compact convex set and let $f : \mathbb{R}^{\xi} \to \mathbb{R}$ be convex and bounded below, with a continuous derivative, ∇f. Furthermore, let $\{x^{(k)}\} \subset \mathcal{X}$ and $\{\hat{x}^{(k)}\} \subset \mathcal{X}$ be generated by* (2.20) *and* (2.21)*, respectively, where $\alpha^{(k)}$ fulfills,*

$$\begin{cases} k\alpha^{(k)} \leq \kappa, \quad \forall k \geq \sigma \\ \alpha^{(k+1)} = \frac{\alpha^{(k)}}{1+\alpha^{(k)}}, \end{cases} \tag{2.23}$$

for some $\sigma > 0$ and $\kappa > 0$, then

$$\lim_{k \to \infty} f(x^{(k)}) = \min_{x \in \mathcal{X}} f(x).$$

Note that $\alpha^{(k)} = \frac{1}{1+k}$ fulfills (2.23).

2.6.3 Center-Free Resource Allocation

Ho et al. (1980) and Xiao and Boyd (2006) consider the problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} & \sum_{i=1}^{n} x_i = x_{\text{tot}} \end{array} \tag{2.24}$$

where the functions $f_i : \mathbb{R} \to \mathbb{R}$ are strictly convex and twice continuously differentiable with the second derivatives fulfilling

$$\mathrm{l}_i \leq f_i''(x_i) \leq \mathrm{u}_i, i = 1, ..., n, \tag{2.25}$$

where $\mathrm{l}_i > 0$ and u_i are known. Ho et al. (1980) have the additional constraints $x_i \geq 0, i = 1, ..., n$, and they propose a peer-to-peer solution algorithm. The algorithm has two parts, an initialization phase and an iterative phase. In the initialization phase, some of the nodes for which the optimal allocation is zero are identified and their allocations are fixed to zero. More specifically, the initialization step identifies the largest set of nodes $\mathcal{I} \subseteq \{1, ..., n\}$ such that

$$\begin{cases} f_i'(x_i) \geq \max_{i \in \mathcal{I}}\{f_i'(0)\}, \forall i \in \mathcal{I} \\ \sum_{i \in \mathcal{I}} x_i = x_{\mathrm{tot}}. \end{cases}$$

Furthermore, Ho et al. (1980) show that $i \notin \mathcal{I}$ implies $x_i^\star = 0$. Only the nodes in the set $\mathcal{I}$ take part in the iterative phase and the other nodes' allocations are set to zero.

In the iterative phase, the nodes exchange resources according to a peer-to-peer algorithm, where the update equation can be written as

$$x^{(k+1)} = x^{(k)} - W \nabla f(x^{(k)}).$$

Xiao and Boyd (2006) focus on (2.24), without the additional non-negativity constraint, and provide more general (than Ho et al. (1980)) sufficient conditions; an algorithm for finding the optimal weights W (similar to (2.10)); and some simple schemes to find a suboptimal W. We will only use a suboptimal W for which sufficient conditions for convergence are

$$\begin{array}{ll} W^\intercal = W, & \begin{cases} W_{ij} \leq 0, & \text{if } (i,j) \in \mathcal{E} \\ W_{ij} = 0, & \text{if } (i,j) \notin \mathcal{E} \text{ and } i \neq j, \end{cases} \\ W\mathbf{1}_n = 0, & \\ W \text{ is irreducible}, & \sum_{j \in \mathcal{N}_i} |W_{ij}| < \mathrm{u}_{\max}, \end{array}$$

where $\mathrm{u}_{\max} = \max_{i=\{1,..,n\}} \mathrm{u}_i$. The condition $\mathbf{1}^\intercal W = 0$ in combination with a feasible initial iterate imply that *all* iterates are feasible (since $\mathbf{1}^\intercal x^{(k+1)} = \mathbf{1}^\intercal x^{(k)} + \mathbf{1}^\intercal W \nabla f(x^{(k)}) = \mathbf{1}^\intercal x^{(k)}$). A simple, but suboptimal, way of guaranteeing convergence is that W should satisfy the following conditions (cf., the Metropolis-Hastings weights (2.9)) (Xiao and Boyd, 2006))

$$\begin{cases} W_{ij} = -\min\left\{\frac{1}{d_i \mathrm{u}_i}, \frac{1}{d_j \mathrm{u}_j}\right\} + \varepsilon, j \in \mathcal{N}(i) \\ W_{ii} = -\sum_{j \in \mathcal{N}(i)} W_{ij} \\ W_{ij} = 0, \text{ otherwise}, \end{cases} \tag{2.26}$$

where ε is a sufficiently small positive constant.

2.7 Decomposition Methods

Decomposition methods are characterized by that the original optimization problem is split into several subproblems. The subproblems are often independent but need to be coordinated to reach an optimal solution to the original problem. This coordination is usually done by having two levels of optimization problems: a master problem that coordinates several subproblems; see Fig. 2.4. We demonstrate some decomposition methods by applying them to the following optimization problem

$$\begin{array}{ll} \underset{x,c}{\text{maximize}} & u(x) \\ \text{subject to} & g(x) \leq c, \quad c \in \mathcal{C} \\ & x \in \mathcal{X}, \end{array} \tag{2.27}$$

where the objective function $u : \mathbb{R}^\xi \to \mathbb{R}$ is concave, the constraint function $g : \mathbb{R}^\xi \to \mathbb{R}^\chi$ is convex, the feasible sets $\mathcal{X} \subseteq \mathbb{R}^\xi$ and $\mathcal{C} \subseteq \mathbb{R}^\chi$ are convex and closed. Hence, we have a convex optimization problem.

The development relies on some basic results on subgradients, which were previously reviewed. Although several surveys over decomposition techniques have been written in the mathematical programming community, e.g., Lasdon (1970), Flippo and Rinnooy Kan (1993), and Holmberg (1995), their focus is typically on exploiting problem structure to improve computational efficiency. In this thesis, our focus is different. Rather than trying to divide the network utility maximization into subproblems that can be solved efficiently (say, in terms of memory or CPU cycles), we use decomposition techniques to divide the optimization of a network-wide performance measure to functions that can be executed in a distributed manner, which fits the structure of the networked system where the optimization problem should be solved. Below, we will review three classes of decomposition principles: primal, dual, and primal-dual. We use primal and dual in their mathematical programming meaning: primal indicates that the optimization problem is solved using the original formulation and variables, while dual indicates that the original problem has been rewritten using Lagrangian relaxation[4].

2.7.1 Dual Decomposition

Dual decomposition is sometimes referred to as *price-directive* decomposition. The name comes from the economic interpretation that the system is directed towards its optimal operation by pricing the common resources. Constraints on the common resource are not explicitly enforced, but the demand is aligned with the supply using a simple pricing strategy: increase the prices on resources that are in shortage and decrease the price of resources that are in excess.

[4]This is in contrast with the literature on congestion control, which we will get back to in Chapter 4, where primal usually implies dynamics in the algorithms at the nodes, and dual usually implies dynamics in the algorithms at the links. For further details, see, e.g., Palomar and Chiang (2006)

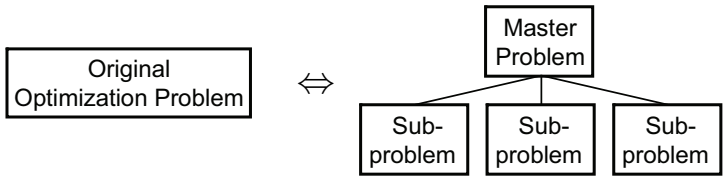


Figure 2.4: The optimization problem is decomposed into two levels: a master problem and several subproblems.

Formally, we apply Lagrange duality to the coupling constraint of (2.27), $g(x) \leq c$, and form the partial Lagrangian[5]

$$\mathrm{L}(x, c, \lambda) = u(x) - \lambda^{\mathsf{T}} g(x) + \lambda^{\mathsf{T}} c, \tag{2.28}$$

where λ are the Lagrange multipliers. The *dual function* is defined as

$$\mathrm{d}(\lambda) = \max_{x \in \mathcal{X},\, c \in \mathcal{C}} \mathrm{L}(x, c, \lambda).$$

Note that $\mathrm{d}(\lambda)$ is separable and can be written as

$$\mathrm{d}(\lambda) = \underbrace{\max_{x \in \mathcal{X}} \{u(x) - \lambda^{\mathsf{T}} g(x)\}}_{\text{Subproblem 1}} + \underbrace{\max_{c \in \mathcal{C}} \{\lambda^{\mathsf{T}} c\}}_{\text{Subproblem 2}}. \tag{2.29}$$

Thus, to evaluate the dual function for a given price vector we need to solve subproblem 1 and subproblem 2.

If $u^\star$ denotes the optimal value of (2.27), then $\mathrm{d}(\lambda) \geq u^\star$, so we can think of $\mathrm{d}(\lambda)$ as an optimistic estimate of the total utility. Intuitively, the coupling constraint is not present as such, but accounted for using the pricing scheme. Since $\mathrm{d}(\lambda)$ is convex, both prices that are too low and too high will yield larger value of $\mathrm{d}(\lambda)$. The optimal prices are obtained by solving the *dual problem*

$$\begin{array}{ll} \underset{\lambda}{\text{minimize}} & \mathrm{d}(\lambda) \\ \text{subject to} & \lambda \geq 0. \end{array} \tag{2.30}$$

The minimization tries to recreate the effect of the constraints $g(x) \leq 0$ on the relaxed problem. Furthermore, denote the optimal value of the dual problem by $\mathrm{d}^\star$; in general, we have that $\mathrm{d}^\star \geq u^\star$. When $\mathrm{d}^\star = u^\star$, we say that *strong duality* holds, and the optimal value of the original problem can be computed via its dual. One condition that guarantees strong duality is Slater's condition, which basically says that if there exists a feasible point where strict inequalities hold (a precise definition

[5] Apparently, Lagrangian was systematically misspelled as Lagrangean in the *Physical Review Letters* in the 1980's (Mermin, 1988). We will try to avoid that error here.

is given below), then strong duality holds. Conditions that guarantee strong duality are often called *constraint qualifications*; we have the following lemma Bertsekas et al. (2003, Proposition 5.4.1).

Lemma 2.7.1. *Consider the following optimization problem*

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & u(x) \\ \textit{subject to} \quad & g(x) \leq 0, \quad h(x) = 0 \\ & x \in \mathcal{X}, \end{aligned} \tag{2.31}$$

where the utility function u is concave, the set $\mathcal{X}$ is convex and closed, the function g is convex, and the function h is affine. If there exists a point $\hat{x}$ in the relative interior of $\mathcal{X}$ such that $h(\hat{x}) = 0$, and there exists a feasible point $\check{x}$ such that $g(\check{x}) < 0$, then strong duality holds.

Remark 2.7.2. If the set $\mathcal{X} \triangleq \mathbb{R}^{\xi}$, then Lemma 2.7.1 reduces to the previously mentioned Slater's condition.

See, e.g., Boyd and Vandenberghe (2004), Bertsekas et al. (2003), Rockafellar (1970), or Luenberger (1969) for more details on Lagrangian duality.

Karush-Kuhn-Tucker Conditions

Lagrange multipliers can also be helpful to characterize the optimal point. Let us consider the problem

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & f(x) \\ \text{subject to} \quad & g(x) \leq 0 \\ & h(x) = 0, \end{aligned} \tag{2.32}$$

where $f : \mathbb{R}^{\xi} \to \mathbb{R}$, $g : \mathbb{R}^{\xi} \to \mathbb{R}^{\chi}$ are differentiable convex functions and $h : \mathbb{R}^{\xi} \to R^{\varrho}$ is an affine function. We have the following lemma (see, e.g., (Boyd and Vandenberghe, 2004, Page 244))

Lemma 2.7.3. *If there is a point $\hat{x}$ such that $g(\hat{x}) < 0$ and a point $\check{x}$ such that $h(\check{x}) = 0$ (Slater's condition), then strong duality holds and the following equations, the so-called Karush-Kuhn-Tucker (KKT) conditions, are necessary and sufficient for an optimal point to the optimization problem* (2.32)*,*

$$g(x^{\star}) \leq 0 \tag{2.33a}$$

$$\lambda^{\star} \geq 0 \tag{2.33b}$$

$$[\lambda^{\star}]_i [g(x^{\star})]_i = 0, \; i = 1, ..., \chi \tag{2.33c}$$

$$h(x^{\star}) = 0 \tag{2.33d}$$

$$\nabla f(x^{\star}) - (\nabla g(x^{\star}))\lambda^{\star} + (\nabla h(x^{\star}))\varsigma^{\star} = 0, \tag{2.33e}$$

where $\lambda^{\star} \in \mathbb{R}^{\chi}$ and $\varsigma^{\star} \in \mathbb{R}^{\varrho}$ are Lagrange multipliers.

Algorithm 3 Dual

Let $k \leftarrow 0$ and $\lambda^{(0)} \in \Lambda$.
loop
Compute a subgradient at $\lambda^{(k)}$ by solving subproblem 1 and subproblem 2.
Update $\lambda^{(k+1)}$ via (2.34) and let $k = k + 1$.
end loop

Solving the Dual Problem

We have the following useful lemma (see, e.g., Bertsekas et al. (2003, section 8.1)).

Lemma 2.7.4. *Let $\lambda \geq 0$ be given, and let x and c be the associated optimal solutions to subproblem 1 and subproblem 2 from* (2.29)*, respectively. Then, a subgradient of* $\mathrm{d}(\lambda)$ *at λ is given by $c - g(x)$.*

Now that we have a subgradient available, we can solve the dual problem using the previously discussed subgradient method. We update the prices according to

$$\lambda^{(k+1)} = \mathcal{P}_{\Lambda}\{\lambda^{(k)} - \alpha^{(k)}(c^{(k)} - g(x^{(k)}))\}, \tag{2.34}$$

and we have the following lemma, which follows from the previous subgradient method discussion.

Lemma 2.7.5. *If a solution to* (2.30) *exists, the subgradients are bounded, and the stepsizes fulfill* (2.19)*, then Algorithm 3 converges to an optimal solution to* (2.30).

There are a number of issues in applying dual decomposition in a distributed setting. The first one is that we often have to resort to diminishing stepsizes to get complete convergence to an optimal solution, using Lemma 2.7.5. The speed of convergence would be better and the implementation would be easier if we could use a fixed stepsize. However, this requires that the dual function is differentiable and Lipschitz continuous. Unfortunately, this is not the case in general, and even if the dual is Lipschitz continuous, the Lipschitz constant is often hard to compute. Another, maybe more critical, issue is that the primal variables obtained for a given price vector λ may not be feasible, even if the dual variables are set to their optimal values (Holmberg, 1995). The following example is a simple case illustrating this phenomena.

Example 2.7.1. Consider the following optimization problem

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & x \\ \text{subject to} \quad & x \leq K, \quad x \in \mathbb{R}. \end{aligned}$$

Strong duality holds and the Lagrangian is $\mathrm{L}(x, \lambda) = x - \lambda(x - K)$. The optimal Lagrange multiplier is $\lambda^\star = 1$ yielding the optimal value $\mathrm{d}(\lambda^\star) = K$. However,

the Lagrangian at $\lambda^\star$, $\mathrm{L}(x, \lambda^\star) = K$, is maximized by an arbitrary x, and, depending on the implementation, we can get primal solutions that are neither optimal nor feasible. However, using the KKT conditions, we get the correct primal solution.

Within linear programming, this property has been referred to as the *non-coordinability* phenomenon. In off-line schemes, this problem can be circumvented if one can devise a method for supplying a feasible primal solution. The following result gives us conditions for convergence of the subproblem solutions.

Lemma 2.7.6. *If the Lagrangian is maximized for a unique s and c for every $\lambda \geq 0$, then the subproblem solutions corresponding to the optimal $\lambda^\star$ are primal optimal.*

Proof. There exist a dual optimal solution $\lambda^\star$, strong duality is assumed to hold, and a primal feasible optimal solution exist. Therefore (Bertsekas et al. (2003, Proposition 6.1.1)), any optimal primal-dual solution must fulfill $\mathrm{L}(s^\star, c^\star, \lambda^\star) = \min_{s\in\mathcal{S}, c\in\mathcal{C}} \mathrm{L}(s, c, \lambda^\star)$. Since this is fulfilled for a unique pair $(s^\star, c^\star)$ they must be primal optimal. □

There are several approaches for attaining primal convergence in dual decomposition schemes. One approach is to add a strictly concave term to the maximization objective, as is done in *proximal point* methods (see, e.g., Bertsekas (1999)). The original problem is then replaced by a sequence of optimization problems

$$\begin{aligned} \underset{x,c}{\text{maximize}} \quad & u(x) - \varepsilon\|c - \tilde{c}\|_2^2 \\ \text{subject to} \quad & g(x) \leq c, \quad c \in \mathcal{C}, \quad \tilde{c} \in \mathbb{R}^\chi, \end{aligned}$$

which is solved one optimization problem at a time. Before solving each optimization problem, we let $\tilde{c} \leftarrow c^\star$, where $c^\star$ is the optimal c for the previous optimization problem in the sequence. The added term makes the dual function smooth, and convergence of the primal variables in the limit follows. For centralized optimization problems, one may also impose primal convergence by solving a master problem (see, e.g., Johansson and Xiao (2006)); however, since the master problem is typically a large-scale convex optimization problem, the approach appears less applicable to distributed implementation. Another alternative is based on forming weighted averages of subproblem solutions in a way that guarantees convergence to the optimal primal variables in the limit (Larsson et al., 1999); however, since the iterates themselves may not be well-behaved, this approach is not always suitable.

2.7.2 Primal Decomposition

Primal decomposition is also called *resource-directive* decomposition. Rather than introducing a pricing scheme for the common resources, the primal decomposition approach sequentially updates the resource allocation to maximize the total utility.

Algorithm 4 Primal

Let $k \leftarrow 0$ and $c^{(0)} \in \mathcal{C}$.
loop
Compute a subgradient at $c^{(k)}$. Solve (2.35) and the optimal dual variables to the corresponding optimization problem is the subgradient.
Update $c^{(k+1)}$ via (2.37) and let $k \leftarrow k + 1$.
end loop

Mathematically, primal decomposition relies on re-writing (2.27) in terms of the *primal function*

$$\nu(c) = \max_{x \in \mathcal{X}} \{u(x) \mid g(x) \leq c\}. \tag{2.35}$$

Note that the primal function is a pessimistic estimate of the achievable utility, since the resource allocation may be fixed at sub-optimal values. The optimal total utility can be found by solving the *primal problem*

$$\begin{array}{ll} \underset{c}{\text{maximize}} & \nu(c) \\ \text{subject to} & c \in \mathcal{C}. \end{array} \tag{2.36}$$

Although the primal function is potentially non-smooth, a subgradient is given by the following lemma (see, e.g., Bertsekas et al. (2003, Section 6.5.3))

Lemma 2.7.7. *Let λ be a vector of optimal dual variables for the constraint $g(x) \leq c$ in the optimization problem corresponding to $\nu(c)$. Assume that the allocated c is such that there exist a strictly feasible x, i.e., $\exists x \in \mathcal{X} : g(x) < c$. A subgradient of $\nu(c)$ at c is given by λ.*

We can thus use the previously discussed subgradient methods to solve the primal problem, updating c using the iteration

$$c^{(k+1)} = \mathcal{P}_{\mathcal{C}}\{c^{(k)} + \alpha^{(k)}\lambda^{(k)}\}. \tag{2.37}$$

We have the following lemma, which follows from the previous subgradient method discussion.

Lemma 2.7.8. *If there exists a solution to* (2.36)*, the subgradients are bounded, and the stepsizes fulfill* (2.19)*, Algorithm 4 converges to an optimal solution to* (2.36)*.*

Contrary to dual decomposition, primal decomposition guarantees that the iterates *always* are feasible by construction.

2.7.3 Primal-Dual Decomposition

In *primal-dual* decomposition schemes, one tries to exploit both primal and dual problem structures. One class of methods, sometimes called *mixed decomposition* applies price- and resource-directive to different components within the same system (Obel, 1978). We will make use of an alternative decomposition scheme, called *cross decomposition* (Van Roy, 1983).

In essence, cross decomposition is an alternating price directive and resource directive decomposition approach. One alternates between the primal and dual subproblems and there is no master problem involved. In general, the pure cross decomposition approach does not converge. However, *mean value cross decomposition* (MVC), where one uses the mean value of all previous solutions, has recently been shown to converge (Holmberg and Kiwiel, 2006). The MVC algorithm, see Algorithm 5, solves the following problem (presented in its original form)

$$\begin{aligned} \underset{x,c}{\text{maximize}} \quad & u(x) + v(c) \\ \text{subject to} \quad & A_1(x) + B_1(c) \leq b_1 \\ & A_2(x) + B_2(c) \leq b_2 \\ & x \in \mathcal{X} \\ & c \in \mathcal{C}, \end{aligned} \tag{2.38}$$

where $u(x)$ and $v(c)$ are concave, $A_1(x), A_2(x), B_1(c)$, and $B_2(c)$ are convex functions, and $\mathcal{X}$ and $\mathcal{C}$ are convex and compact sets. Let the optimal value be $u^\star + v^\star$. It is also assumed that for any fixed $c \in \mathcal{C}$, there exists a point $x \in \mathcal{X}$ such that

$$A_1(x) + B_1(c) < b_1 \text{ and } A_2(x) + B_2(c) < b_2,$$

implying that strong duality holds for the two coupling constraints. Define the partial Lagrangian as

$$\mathrm{L}(x, c, \lambda) = u(x) + v(c) - \lambda^\intercal (A_1(x) + B_1(c) - b_1),$$

and define $\kappa(\bar{c}, \bar{\lambda})$ for any $\bar{c} \in \mathcal{C}$ and $\bar{\lambda} \geq 0$ as

$$\kappa(\bar{c}, \bar{\lambda}) = \max_{x \in \mathcal{X}} \left\{ \mathrm{L}(x, \bar{c}, \bar{\lambda}) \middle| A_2(x) + B_2(\bar{c}) \leq b_2 \right\}.$$

The primal subproblem is defined as

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \quad & \kappa(\bar{c}, \lambda) \\ \text{subject to} \quad & \lambda \geq 0, \end{aligned}$$

and the dual subproblem is defined as

$$\begin{aligned} \underset{c}{\text{maximize}} \quad & \kappa(c, \bar{\lambda}) \\ \text{subject to} \quad & c \in \mathcal{C}. \end{aligned} \tag{2.39}$$

Algorithm 5 MVC

1: Let $\bar{c}^{(0)} \in \mathcal{C}$, $\bar{\lambda}^{(0)} \geq 0$, and $k \leftarrow 0$.
2: **loop**
3: Solve the primal subproblem (2.40) for $\bar{c}^{(k)}$ to get $\lambda^{(k+1)}$.
4: Solve the dual subproblem (2.39) for $\bar{\lambda}^{(k)}$ to get $c^{(k+1)}$.
5: Update the averages with (2.41) and let $k \leftarrow k+1$.
6: **end loop**

By strong duality (applicable by assumption), the primal subproblem can be rewritten as

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & u(x) + v(\bar{c}) \\ \text{subject to} \quad & A_1(x) + B_1(\bar{c}) \leq b_1 \\ & A_2(x) + B_2(\bar{c}) \leq b_2 \\ & x \in \mathcal{X}. \end{aligned} \tag{2.40}$$

The primal and dual subproblem are solved alternatingly for the mean values of all previous iterations, where the mean values are defined as

$$\bar{\lambda}^{(k)} = \sum_{i=1}^{k} \frac{\lambda^{(i)}}{k} \text{ and } \bar{c}^{(k)} = \sum_{i=1}^{k} \frac{c^{(i)}}{k}, \tag{2.41}$$

and we have the following lemma for the these averages.

Lemma 2.7.9 (Holmberg and Kiwiel (2006, Theorem 1))**.** *Under the assumptions given, Algorithm 5 converges to the optimal solution to* (2.38)*, i.e.,*

$$\lim_{k\to\infty} \operatorname*{dist}_{\mathcal{C}^\star}(\bar{c}^{(k)}) = 0 \text{ and } \lim_{k\to\infty} \operatorname*{dist}_{\Lambda^\star}\big(\bar{\lambda}^{(k)}\big) = 0,$$

with

$$\mathcal{C}^\star = \{c \in \mathcal{C} | \min_{\lambda\geq 0} \kappa(c,\lambda) = u^\star + v^\star\},$$

$$\Lambda^\star = \{\lambda \geq 0 | \max_{c\in\mathcal{C}} \kappa(c,\lambda) = u^\star + v^\star\},$$

and $\operatorname{dist}_{\mathcal{X}}(x) \triangleq \inf\{\|x-z\| \mid z \in \mathcal{X}\}$.

2.7.4 Saddle-Point Computations and Min-Max Games

An alternative path to finding primal-dual optimal solutions to (2.27) goes via the so-called saddle-point characterization of optimal points.

A *saddle point* for a function $f : \mathbb{R}^\xi \times \mathbb{R}^\chi \mapsto \mathbb{R}$ is any pair $(\tilde{w}, \tilde{z})$ such that

$$f(\tilde{w}, z) \leq f(\tilde{w}, \tilde{z}) \leq f(w, \tilde{z})$$

holds for all feasible w, z, i.e.,

$$f(\tilde{w}, \tilde{z}) = \max_z f(\tilde{w}, z) \text{ and } f(\tilde{w}, \tilde{z}) = \min_w f(w, \tilde{z}).$$

This implies that the *strong max-min property*

$$\max_z \min_w f(w, z) = \min_w \max_z f(w, z)$$

holds (and that the common value is $f(\tilde{w}, \tilde{z})$).

By weak duality, we have

$$\max_{x \in \mathcal{X},\, c \in \mathcal{C}} \min_{\lambda \geq 0} \mathrm{L}(x, c, \lambda) \leq u^\star \leq \min_{\lambda \geq 0} \max_{x \in \mathcal{X},\, c \in \mathcal{C}} \mathrm{L}(x, c, \lambda).$$

This inequality, known as the *max-min inequality*, simply states that the primal problem underestimates the optimal value, while the dual problem overestimates it. Under strong duality, the inequality holds with equality as the primal and dual optimal values are equal. The optimal point can then be given the following alternative characterization: $(x^\star, c^\star, \lambda^\star)$ is a primal-dual optimal point to the optimization problem if and only if $(x^\star, c^\star) \in \mathcal{X} \times \mathcal{C}$, $\lambda^\star \geq 0$ and $(x^\star, c^\star, \lambda^\star)$ forms a saddle point of the Lagrangian, in the sense that

$$\mathrm{L}(x, c, \lambda^\star) \leq \mathrm{L}(x^\star, c^\star, \lambda^\star) \leq \mathrm{L}(x^\star, c^\star, \lambda)$$

for all $(x, c) \in \mathcal{X} \times \mathcal{C}$, $\lambda \geq 0$ (cf., Bertsekas (1999, Proposition 5.1.6)). In other words, $\lambda^\star$ minimizes $L(x^\star, c^\star, \lambda)$ while $(x^\star, c^\star)$ maximizes $\mathrm{L}(x, c, \lambda^\star)$.

Moreover, for (2.27) we have that if (x, c, λ) form a saddle point of the Lagrangian, then (x, c) are primal optimal and λ is dual optimal, without any convexity assumptions on (2.27) (Lasdon, 1970, Theorem 2, page 85).

One of the most well-known algorithms for finding saddle points is the algorithm due to Arrow-Hurwicz (Arrow et al., 1958)

$$\begin{aligned}
x^{(k+1)} &= \mathcal{P}_{\mathcal{X}} \left\{ x^{(k)} + \alpha^{(k)} \nabla_x \mathrm{L}(x^{(k)}, c^{(k)}, \lambda^{(k)}) \right\} \\
c^{(k+1)} &= \mathcal{P}_{\mathcal{C}} \left\{ c^{(k)} + \alpha^{(k)} \nabla_c \mathrm{L}(x^{(k)}, c^{(k)}, \lambda^{(k)}) \right\} \\
\lambda^{(k+1)} &= \mathcal{P}_{\Lambda} \left\{ \lambda^{(k)} - \alpha^{(k)} \nabla_\lambda \mathrm{L}(x^{(k)}, c^{(k)}, \lambda^{(k)}) \right\},
\end{aligned}$$

where $\alpha^{(k)}$ is the stepsize. Although the iteration is not guaranteed to converge (unless one imposes the additional requirements of strict convexity-concavity (Nemirovski and Judin, 1978)), it provides a unified view of the primal and dual decomposition methods. In particular, the decomposition schemes can be interpreted as methods that run the above updates on different time-scales. Primal decomposition lets the x and λ dynamics run on a fast time-scale (essentially, until convergence) while the resource updates, c, are run on a slow time-scale. Similarly, dual decomposition can be seen as letting the x and c updates run on a fast time-scale, while the λ variables are updated slowly.

Further insight into the decomposition schemes can be gained from the following so-called zero-sum game interpretation of the max-min inequality: consider a game where the dual player selects λ, while the primal player picks s, c and collects $\mathrm{L}(x, c, \lambda)$ dollar from the dual player (i.e., the primal player gets what the dual player loses, hence a zero-sum game). If the dual player goes first, he will try to minimize the amount that he can be forced to pay, i.e., he will let $\lambda = \arg\min_{\lambda \geq 0}\{\max_{x \in \mathcal{X}, c \in \mathcal{C}} \mathrm{L}(x, c, \lambda)\}$ resulting in the payoff $\bar{u}$. Conversely, if the primal player goes first, he will try to maximize the amount that the dual player is guaranteed to pay and thus let $(x, c) = \arg\max_{x \in \mathcal{X}, c \in \mathcal{C}}\{\min_{\lambda \geq 0} \mathrm{L}(x, c, \lambda)\}$, leading to the payoff $\underline{u}$. The max-min inequality simply states that it is best to go second, $\bar{u} \geq \underline{u}$. In convex games with strong duality there is no advantage to go second, since the inequality holds with equality. These min-max and max-min solutions are *Nash equilibriums*.

The mean value cross decomposition can be seen as a repeated zero-sum game where the dual and primal players act alternatingly. During the course of the game, the players remember earlier moves and decide their own strategy under the assumption that the other will use the average strategy over the history of the game.

2.8 Summary

In this chapter, we have introduced important definitions and notions. We also reviewed some standard optimization methods as well as a few more exotic optimization methods, namely, the conditional gradient method with open-loop stepsizes, the mean value cross decomposition method, and the center-free resource allocation method. Finally, we also reviewed some well-known decomposition techniques.

Chapter 3

Novel Distributed Optimization Algorithms

"*...the 'dragon' of optimization is multiheaded and it takes a special sword to cut-off each head.*"

V. F. Dem'yanov and L. V. Vasilev, Nondifferentiation Optimization, 1985.

Distributed algorithms, and distributed optimization algorithms in particular, can be useful from several viewpoints. Such optimization algorithms are typically used for speed and numerical efficiency. On the other hand, in this thesis, we are primarily interested in such optimization algorithms to be able to easily implement them in a networked system to let the subsystems jointly solve an optimization problem, e.g., resource allocation or maximum likelihood estimation. Furthermore, large-scale networked systems are becoming ubiquitous and increasingly complex to manage. If the desired behavior of the networked system can be formulated as an optimization problem, then complexity issues can be mitigated by solving the resulting optimization problem with a distributed algorithm. In this chapter, we develop such distributed algorithms.

The specific outline of the chapter is as follows. We start with some general background in Section 3.1. Then, we will consider two problems: first, we look at a coupled non-smooth convex optimization problem, which is rather general with many applications. This problem is described in Section 3.2. We then devise two different algorithms that solve this problem in Section 3.3 and Section 3.4. The performance of the two algorithms is assessed using numerical experiments in Section 3.5. Second, in Section 3.6, we develop an optimization algorithm for a non-smooth resource allocation problem with a global resource constraint, which extends the center-free resource allocation algorithm in Section 2.6.3. In Section 3.7, we explore the performance of this algorithm using numerical experiments. Finally, we summarize our findings and conclude with Section 3.8.

3.1 Background

There is a substantial interest in distributed optimization algorithms, since centralized algorithms scale poorly with the number of nodes and are less resilient to single node failure (especially sensitive to the failure of the central node). Moreover, peer-to-peer algorithms, which only exchange data between immediate neighbors, are attractive, since they make minimal assumptions on the networking support required for message passing between nodes. Application examples include estimation in sensor networks, coordination in multi-agent systems, and resource allocation in wireless systems; see, e.g., Johansson et al. (2008*e*) and Rabbat and Nowak (2004).

A general and thorough background on distributed and parallel algorithms can be found in Bertsekas and Tsitsiklis (1997). Distributed algorithms are also frequently used in other areas than optimization; Lynch (1996) provides a computer science perspective.

We will consider subgradient based algorithms, which is a class of algorithms that enjoy wide applicability and do not require many assumptions on the objective functions (convexity is the most crucial required assumption). The drawbacks are that these methods can be rather slow and that it is hard to guarantee progress at each iteration. But subgradient based methods are very simple, at least from a mathematical point of view, to implement. We will discuss our experiences from implementing some of these algorithms in a real networked system in Chapter 6).

The workhorse of distributed optimization is most often Lagrangian relaxation, where the optimization problem is decoupled into smaller subproblems and a master problem; see Section 2.7. However, we will not use this approach in this chapter. Other fundamentally different approaches to distributed optimization, typically based on heuristic arguments, include particle swarm optimization and parallel genetic algorithms; see, e.g., van Ast et al. (2008) and Alba and Troya (1999).

3.2 Non-smooth Coupled Convex Optimization

We will now consider the following convex optimization problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \sum_{i=1}^{n} f_i(x) \\ \text{subject to} & x \in \mathcal{X}, \end{array} \tag{3.1}$$

where $f_i : \mathcal{X} \to \mathbb{R}$ are convex functions and $\mathcal{X} \subseteq \mathbb{R}^\xi$ is a convex and closed set. Let $f(x) = \sum_{i=1}^{n} f_i(x)$; the optimal value and the optimizer of (3.1) are denoted $f^\star$ and $x^\star$, respectively. To the problem we associate a connected n-node network, specified by the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We can now interpret the problem as a *networked system*, where each node incurs a loss $f_i(x)$ of operating at x and nodes cooperate with their neighbors to find the optimal operating point (the optimizer $x^\star$ of (3.1)). In other words, each component in the objective function corresponds to a node in a network; see Fig. 3.1 for an example setup. We make the following assumptions.

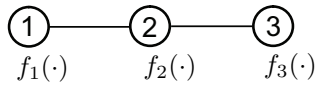


Figure 3.1: Example setup with three nodes. A line between nodes implies that they are connected and that they can communicate with each other.

Assumption 3.2.1. *i) The functions $f_i : \mathcal{X} \to \mathbb{R}$ are convex and the set $\mathcal{X} \subseteq \mathbb{R}^\xi$ is closed, convex, and non-empty. ii) The subgradients are bounded,*

$$\sup \{\|z\| \mid z \in \partial f_i(x),\, i \in 1, ..., n,\, x \in \mathcal{X}\} \leq \varphi,$$

with $\varphi > 0$.

Remark 3.2.2. Two important examples where the subgradients are bounded are: Firstly, the functions f_i are the pointwise maximum of a finite set of linear functions. Secondly, the set $\mathcal{X}$ is compact.

Assumption 3.2.3. *The undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is composed of n nodes and is connected.*

The last assumption guarantees that there is a path between any pair of nodes.

Several important problems in applications such as resource allocation in computer networks (Chiang et al., 2007; Johansson et al., 2006*b*), estimation in sensor networks (Rabbat et al., 2005), and the rendezvous problem in multi-agent systems (Johansson et al., 2008*e*), can be posed as coupled optimization problems. In these problems, each node or agent in the network is associated with a component of the objective function, which depends on a network-wide decision variable.

3.3 Markov Incremental Subgradient Method

In this section, we devise and analyze a novel distributed algorithm that iteratively solves (3.1) by passing an estimate of the optimizer between *neighboring nodes* in the network.

One popular way of solving (3.1), is to use subgradient methods, which were pioneered by Shor (1964) and Ermol'ev (1966); see Shor (1985) for an early summary. These methods have recently been unified in Kiwiel (2004), where an extensive reference list also can be found. The methods' popularity stems from their ease of implementation and their capability of handling non-differentiable objective functions. Another key property is that subgradient methods often can be executed in a distributed fashion. The prototype subgradient method iteration for constrained convex minimization is

$$x^{(k+1)} = \mathcal{P}_{\mathcal{X}} \left[x^{(k)} - \alpha^{(k)} a^{(k)}(x^{(k)}) \right], \tag{3.2}$$

where $\alpha^{(k)}$ is a stepsize, and $a^{(k)}(x^{(k)})$ is a subgradient of the objective function at $x^{(k)}$; there exist quite a few variations and extensions, but none of them fit our needs.

Naturally, the structure of the problem can be exploited and tailored algorithms, e.g., so-called *incremental subgradient methods*, can be used. These algorithms are based on the iteration

$$x^{(k+1)} = \mathcal{P}_{\mathcal{X}} \left[x^{(k)} - \alpha^{(k)} a_{w^{(k)}}(x^{(k)}) \right], \tag{3.3}$$

where $a_{w^{(k)}}(x^{(k)})$ is a subgradient of the function $f_{w^{(k)}}$ at $x^{(k)}$. This type of algorithm was pioneered by Kibardin (1980). Depending on how $w^{(k)}$ and $\alpha^{(k)}$ are chosen, the resulting algorithms have rather diverse properties; the stepsize, $\alpha^{(k)}$, typically needs to be diminishing to insure asymptotic convergence of the iterates to an optimizer. To the author's knowledge, most results on deterministic incremental subgradient methods are covered and unified in Kiwiel (2004). Although these methods were originally devised to boost convergence speed, they can also be used as decentralized mechanisms for optimization. A simple decentralized algorithm, proposed and analyzed in Nedić and Bertsekas (2001), is to use (3.3) with a fixed stepsize and let $w^{(k)}$ cycle deterministically over the set $\{1, ..., n\}$ in a round-robin fashion. We call this algorithm the *deterministic incremental subgradient method* (DISM). In Nedić and Bertsekas (2001), another variant, also suitable for distributed implementation, is proposed: it is a randomized algorithm where $w^{(k)}$ is a sequence of independent and identically distributed random variables which take on values from the set $\{1, ..., n\}$ with equal probability. We call this algorithm the *randomized incremental subgradient method* (RISM). If the problem setup permits, it is also possible to use incremental gradient methods (Blatt et al., 2007). In all of these methods, the iterate can be interpreted as being passed between the nodes in the network. Finally, the iteration (3.3) is similar to the iterations used in stochastic approximation (Kushner and Yin, 2003). However, in stochastic approximation algorithms, the stepsize is typically diminishing and *not* fixed as it is in the algorithm we propose and analyze in this section.

We will now develop an algorithm that is based on (3.3) where the sequence $w^{(k)}$ is constructed such that *only neighbors* need to communicate with each other (in techspeak, this means that the network does not need to provide any multi-hop routing mechanism for the message passing). This is in contrast with both RISM and DISM, where nodes far apart need to communicate with each other.

The following sections constitute a refined version of our previous work Johansson et al. (2007) and Johansson et al. (2008*d*). We note that there is similar work, namely Ram et al. (2008), which is also based on Johansson et al. (2007), that has been conducted in parallel with our developments. Ram et al. (2008) consider a similar algorithm, which is based on a *non*-homogeneous Markov chain and allows for stochastic errors in the subgradient; they analyze diminishing stepsizes as well as constant stepsizes. As they point out, it is not possible to directly compare the resulting bounds, since they assume a more general model, which results in different

and weaker bounds.

In Section 3.3.1, we present the novel algorithm, and in Section 3.3.2 we analyze its convergence properties. Then, in Section 3.3.3, we compare, in the sense of performance bounds, the novel algorithm with the existing algorithms DISM and RISM.

3.3.1 Algorithm

We associate an n-state homogeneous Markov chain, *MC*, with the optimization problem (3.1). We make the following assumptions.

Assumption 3.3.1. *The Markov chain MC is irreducible, aperiodic, and its stationary distribution is uniform.*

We are now ready to define our novel algorithm, which we denote the *markov incremental subgradient method* (MISM). The iterations are defined as follows

$$x^{(k+1)} = \mathcal{P}_{\mathcal{X}}\left[x^{(k)} - \alpha a_{w^{(k)}}(x^{(k)})\right], \; k \geq 0, \tag{3.4}$$

where $w^{(k)} \in \{1, ..., n\}$ is the state of MC.

Remark 3.3.2. The generalization (3.4) is interesting in its own right, but it is particularly interesting in the context of distributed implementation: if the graph $\mathcal{G}$ is connected and MC fulfills the constraints $[P]_{ij} = 0$ when $(i,j) \notin \mathcal{E}$, then $x^{(k)}$ can be interpreted and implemented as an estimate of the optimizer that is passed around in the network between neighboring nodes and thereby iteratively improved. The sparsity constraints guarantee that the state of MC only can jump from state i to state j in one time tick if $(i,j) \in \mathcal{E}$. Furthermore, as we saw in Section 2.4, MC can be constructed using *only local topology information*.

3.3.2 Convergence Analysis

To show convergence we need some notation and three lemmas.

Technical Preliminaries

Denote the starting state of MC by i. Under Assumption 3.3.1 and with probability 1, all states in MC are visited infinitely often. Thus, we can form the subsequence $\{\hat{x}^{(k)}\}_{k=0}^{\infty}$ of $\{x^{(l)}\}_{l=0}^{\infty}$ by sampling it whenever the Markov chain visits the state i, i.e., $w^{(l)} = i$. For example, if $w^{(0)} = i$, $w^{(3)} = i$, and $w^{(5)} = i$, then

$$\begin{array}{cccccc} w^{(0)}, & w^{(1)}, w^{(2)}, w^{(3)}, & w^{(4)}, w^{(5)}, & ... \\ \boxed{x^{(0)}}, & \underbrace{x^{(1)}, x^{(2)}, \boxed{x^{(3)}}}_{R_i^{(0)}}, & \underbrace{x^{(4)}, \boxed{x^{(5)}}}_{R_i^{(1)}}, & ... \\ \hat{x}^{(0)}, & \hat{x}^{(1)}, & \hat{x}^{(2)}, & ..., \end{array} \tag{3.5}$$

where $\hat{x}^{(0)} = x^{(0)}$, $\hat{x}^{(1)} = x^{(3)}$, $\hat{x}^{(2)} = x^{(5)}$. In addition, let $R_i^{(k)}$ be the recurrence time for state i,

$$R_i^{(k)} = \begin{cases} \inf\left\{t - \sum_{m=0}^{k-1} R_i^{(m)} \,|\, w^{(t)} = i,\, t \geq \sum_{m=0}^{k-1} R_i^{(m)} + 1,\, t \in \mathbb{N}\right\},\, k > 0 \\ \inf\left\{t \,|\, w^{(t)} = i,\, t \geq 1,\, t \in \mathbb{N}\right\},\, k = 0 \\ 0,\, k < 0. \end{cases} \tag{3.6}$$

Successive recurrence times to a state form an independent and identically distributed sequence of random variables, due to the strong Markov property (see Section 2.4), and we note that the statistics of $R_i^{(k)}$ will not depend on k. Also note that $R_i^{(k)}$ is independent of $\{\hat{x}^{(k)}, \hat{x}^{(k-1)}, \hat{x}^{(k-2)}, ...\}$. Furthermore, we let $v_{i,j}^{(k)}$ be the random number of visits to state j during the time interval $\left[\sum_{m=0}^{k-1} R_i^{(m)} + 1, \sum_{m=0}^{k} R_i^{(m)}\right]$.

The first lemma concerns the average number of visits to other states over a recurrence time.

Lemma 3.3.3. *Under Assumption 3.3.1, we have that*

$$\left(\mathbb{E}[v_{i,1}^{(k)}] \quad \dots \quad \mathbb{E}[v_{i,n}^{(k)}]\right) = \mathbf{1}_n^\intercal \text{ and } \mathbb{E}\left[R_i^{(k)}\right] = n, \text{ for all } i = 1, ..., n, \; k \geq 0.$$

Proof. Note that the statistics of $v_{i,1}^{(k)}$ and $R_i^{(k)}$ do not depend on k due to the strong Markov property. From Norris (1998, Theorem 1.7.5), we have that

$$\left(\mathbb{E}[v_{i,1}^{(k)}] \quad \dots \quad \mathbb{E}[v_{i,n}^{(k)}]\right)P = \left(\mathbb{E}[v_{i,1}^{(k)}] \quad \dots \quad \mathbb{E}[v_{i,n}^{(k)}]\right).$$

Furthermore, we also know that the transition matrix P has only one eigenvector with eigenvalue 1, namely the invariant distribution (Kemeny and Snell, 1960, Theorem 4.1.6). Since P is assumed to have a uniform stationary distribution, we have that $\left(\mathbb{E}[v_{i,1}^{(k)}] \quad \dots \quad \mathbb{E}[v_{i,n}^{(k)}]\right) = \mathbf{1}_n^\intercal$, which is the desired result. Finally, $\mathbb{E}\left[R_i^{(k)}\right] = \sum_{j=1}^{n} \mathbb{E}[v_{i,j}^{(k)}] = n$. □

The next lemma concerns the second moment of the recurrence times, $\mathbb{E}\left[(R_i^{(k)})^2\right]$.

Lemma 3.3.4 (Kemeny and Snell (1960, Theorem 4.5.2))**.** *Under Assumption 3.3.1, the second moment of the recurrence time $R_i^{(k)}$ is finite and given as*

$$\mathbb{E}\left[\left(R_i^{(k)}\right)^2\right] = 2[\Gamma]_{ii} n^2 - n,$$

with $\Gamma = \left(I - P + \lim_{k\to\infty} P^k\right)^{-1}$.

The last lemma concerns a bounding inequality that we will use in the convergence proof.

Lemma 3.3.5. *Under Assumptions 3.2.1 and 3.3.1, the sequence* $\{\hat{x}^{(k)}\}_{k=0}^{\infty}$*, formed by sampling the sequence* $\{x^{(l)}\}_{l=0}^{\infty}$*, which is generated by* (3.4)*, whenever* $w^{(l)} = i$*, fulfills*

$$\mathbb{E}\left[\left\|\hat{x}^{(k+1)} - y\right\|^2 \middle| \hat{x}^{(k)}\right] \leq \left\|\hat{x}^{(k)} - y\right\|^2 - 2\alpha\left(f\left(\hat{x}^{(k)}\right) - f\left(y\right)\right) + \alpha^2\varphi^2\kappa, \quad (3.7)$$

with $\kappa = \max_i \mathbb{E}\left[\left(R_i^{(k)}\right)^2\right] < \infty$.

Proof. In this proof, we need to use both sequences $\{\hat{x}^{(k)}\}_{k=0}^{\infty}$ and $\{x^{(l)}\}_{l=0}^{\infty}$, and we need to keep track of which elements correspond to each other. For this purpose, let $l = \sum_{m=0}^{k-1} R_i^{(m)}$, so that $x^{(l)} = \hat{x}^{(k)}$ and $x^{(l+R_i^{(k)})} = \hat{x}^{(k+1)}$. Using the non-expansion property of Euclidean projection, the definition of a subgradient, and the assumption that the subgradients are bounded, we have that, for any $y \in \mathcal{X}$,

$$\begin{aligned}\left\|x^{(l+1)} - y\right\|^2 &\leq \left\|x^{(l)} - y\right\|^2 - 2\alpha\left(a_{w^{(l)}}\left(x^{(l)}\right)\right)^{\mathsf{T}}\left(x^{(l)} - y\right) + \alpha^2\varphi^2 \\ &\leq \left\|x^{(l)} - y\right\|^2 - 2\alpha\left(f_{w^{(l)}}\left(x^{(l)}\right) - f_{w^{(l)}}\left(y\right)\right) + \alpha^2\varphi^2.\end{aligned}$$

Along the same lines of reasoning, we get the family of inequalities

$$\begin{aligned}\left\|x^{(l+1)} - y\right\|^2 &\leq \left\|x^{(l)} - y\right\|^2 - 2\alpha\left(f_{w^{(l)}}\left(x^{(l)}\right) - f_{w^{(l)}}\left(y\right)\right) + \alpha^2\varphi^2, \\ \left\|x^{(l+2)} - y\right\|^2 &\leq \left\|x^{(l+1)} - y\right\|^2 - 2\alpha\left(f_{w^{(l+1)}}\left(x^{(l+1)}\right) - f_{w^{(l+1)}}\left(y\right)\right) + \alpha^2\varphi^2, \\ &\vdots \\ \left\|x^{(l+R_i^{(k)})} - y\right\|^2 &\leq \left\|x^{(l+R_i^{(k)}-1)} - y\right\|^2 \\ &\quad - 2\alpha\left(f_{w^{\left(l+R_i^{(k)}-1\right)}}\left(x^{(l+R_i^{(k)}-1)}\right) - f_{w^{\left(l+R_i^{(k)}-1\right)}}\left(y\right)\right) + \alpha^2\varphi^2.\end{aligned}$$

Combining all of them together we get

$$\left\|x^{(l+R_i^{(k)})} - y\right\|^2 \leq \left\|x^{(l)} - y\right\|^2 - 2\alpha\sum_{j=0}^{R_i^{(k)}-1}\left(f_{w^{(l+j)}}\left(x^{(l+j)}\right) - f_{w^{(l+j)}}\left(y\right)\right) + R_i^{(k)}\alpha^2\varphi^2,$$

which can be rewritten as

$$\left\|x^{(l+R_i^{(k)})}-y\right\|^2 \leq \left\|x^{(l)}-y\right\|^2 - 2\alpha \sum_{j=0}^{R_i^{(k)}-1}\left(f_{w^{(l+j)}}\left(x^{(l+j)}\right)-f_{w^{(l+j)}}\left(x^{(l)}\right)\right) - 2\alpha \sum_{j=0}^{R_i^{(k)}-1}\left(f_{w^{(l+j)}}\left(x^{(l)}\right)-f_{w^{(l+j)}}(y)\right)+R_i^{(k)}\alpha^2\varphi^2. \quad (3.8)$$

Notice that

$$\begin{aligned} f_{w^{(l+j)}}\left(x^{(l)}\right)-f_{w^{(l+j)}}\left(x^{(l+j)}\right) &\leq \left\|a_{w^{(l+j)}}(x^{(l)})\right\|\left\|x^{(l+j)}-x^{(l)}\right\| \\ &\leq \varphi\left\|x^{(l+j)}-x^{(l)}\right\| \leq \alpha j\varphi^2. \end{aligned}$$

This enables us to re-write inequality (3.8) as follows

$$\left\|x^{(l+R_i^{(k)})}-y\right\|^2 \leq \left\|x^{(l)}-y\right\|^2 - 2\alpha \sum_{j=0}^{R_i^{(k)}-1}\left(f_{w^{(l+j)}}\left(x^{(l)}\right)-f_{w^{(l+j)}}(y)\right)+\alpha^2\varphi^2\left(R_i^{(k)}\right)^2.$$

Using $v_{i,j}^{(k)}$ as defined in Lemma 3.3.3, we express (3.8) as

$$\left\|x^{(l+R_i^{(k)})}-y\right\|^2 \leq \left\|x^{(l)}-y\right\|^2 - 2\alpha\sum_{j=1}^{n} v_{i,j}^{(k)}\left(f_j\left(x^{(l)}\right)-f_j(y)\right)+\alpha^2\varphi^2\left(R_i^{(k)}\right)^2. \quad (3.9)$$

Now, due to the Markov property and Lemma 3.3.3, we have

$$\mathbb{E}\left[\sum_{j=1}^{n} v_{i,j}^{(k)}\left(f_j\left(x^{(l)}\right)-f_j(y)\right)\middle| x^{(l)}, w^{(l)}\right] = \sum_{j=1}^{n}\mathbb{E}[v_{i,j}^{(k)}]\left(f_j\left(x^{(l)}\right)-f_j(y)\right) = f\left(x^{(l)}\right)-f(y). \quad (3.10)$$

Define

$$\kappa = \max_{j\in\{1,\ldots,n\}}\mathbb{E}\left[\left(R_j^{(k)}\right)^2\right], \quad (3.11)$$

which is known to be finite and easily computable[1] from Lemma 3.3.4. Note that $\kappa \geq \mathbb{E}\left[\left(R_j^{(k)}\right)^2\right] = \mathbb{E}\left[\left(R_j^{(k)}\right)^2 \mid x^{(l)}, w^{(l)}\right]$ for any $j \in \{1, ..., n\}$. By taking the conditional expectation of (3.9) with respect to $x^{(l)}$ and $w^{(l)}$, and using the equations (3.10) and (3.11), we obtain the desired result. □

Proof of Convergence

Now we are ready for the main result of this section.

Theorem 3.3.6. *Let $\{x^{(l)}\}_{l=0}^{\infty}$ be generated by (3.4). Under Assumption 3.2.1, Assumption 3.3.1, and with probability 1, we have the following:*

a) The sequence $\{x^{(l)}\}_{l=0}^{\infty}$ fulfills

$$\begin{cases} \liminf_{l\to\infty} f\left(x^{(l)}\right) = f^\star, & \text{if } f^\star = -\infty \\ \liminf_{l\to\infty} f\left(x^{(l)}\right) \leq f^\star + \frac{\alpha\varphi^2\kappa}{2}, & \text{if } f^\star > -\infty. \end{cases}$$

b) If the set of all optimal x, $\mathcal{X}^\star = \{x \in \mathcal{X} | f(x) = f^\star\}$, is non-empty, then the sequence $\{x^{(l)}\}_{l=0}^{\infty}$ fulfills

$$\min_{0\leq l\leq \tau} f\left(x^{(l)}\right) \leq f^\star + \frac{\alpha\varphi^2\kappa}{2} + \delta,$$

where τ is a stopping time with bounded expected value

$$\mathbb{E}[\tau] \leq \frac{n}{2\alpha\delta}\left(\operatorname*{dist}_{\mathcal{X}^\star}(x^{(0)})\right)^2,$$

with $\operatorname{dist}_{\mathcal{X}^\star}(x^{(0)}) = \inf\{\|x^{(0)} - y\| | y \in \mathcal{X}^\star\}$.

Proof. We begin with showing a). Denote the starting state of MC by i. With probability 1, all states are visited equally often, and thus we can form the subsequence $\{\hat{x}^{(k)}\}_{k=0}^{\infty}$ of $\{x^{(l)}\}_{l=0}^{\infty}$ by sampling it whenever MC visits the state i; see (3.5) for an illustration of this sampling.

We attack the problem using an approach similar to that of the proof of Proposition 3.1 in Nedić and Bertsekas (2001). The proof idea is to show that the iterates will always enter a special level set. For this purpose, let ϱ and θ be positive integers. If $\sup_{x\in\mathcal{X}} f(x) < f^\star + \frac{1}{\varrho}$, then the iterates trivially fulfill

$$x^{(l)} \in \left\{ x \in \mathcal{X} \,\middle|\, f(x) < f^\star + \frac{1}{\varrho} \right\}, l \geq \theta.$$

[1]The constant κ is easily computable in a centralized fashion if P is known. The transition matrix P is usually not known by any node in the setups we consider, but κ is only needed for the theoretical performance bounds, and it is *not* needed to execute the algorithm.

Otherwise, if $\sup_{x\in\mathcal{X}} f(x) \geq f^\star + \frac{1}{\varrho}$, we can let $y_\varrho \in \mathcal{X}$ be such that

$$f\left(y_\varrho\right) = \begin{cases} -\psi, & \text{if } f^\star = -\infty \\ f^\star + \frac{1}{\varrho}, & \text{if } f^\star > -\infty, \end{cases}$$

for some $\psi \geq \varrho$. We now introduce the special level set $\mathcal{L}_\varrho$, defined by

$$\mathcal{L}_\varrho = \left\{ x \in \mathcal{X} \middle| f(x) \leq f\left(y_\varrho\right) + \frac{1}{\varrho} + \frac{\alpha\varphi^2\kappa}{2} \right\}.$$

Note that this set includes y_ϱ. To simplify the analysis, we derive a stopped sequence from $\{\hat{x}^{(k)}\}_{k=\theta}^{\infty}$ by defining the sequence $\{\tilde{x}^{(k)}\}_{k=\theta}^{\infty}$ as follows

$$\tilde{x}^{(k)} = \begin{cases} \hat{x}^{(k)} & \text{if } \hat{x}_j \notin \mathcal{L}_\varrho \ \forall j \in \{\theta, ..., \kappa\} \\ y_\varrho & \text{otherwise.} \end{cases}$$

When $\tilde{x}^{(k)} \notin \mathcal{L}_\varrho$, by setting $y = y_\varrho$ in (3.7) in Lemma 3.3.5, we get

$$\mathbb{E}\left[\left\| \tilde{x}^{(k+1)} - y_\varrho \right\|^2 \middle| \tilde{x}^{(k)}, w^{(k)} \right] \leq \left\| \tilde{x}^{(k)} - y_\varrho \right\|^2 + \alpha^2\varphi^2\kappa - 2\alpha \left(f\big(\tilde{x}^{(k)}\big) - f\left(y_\varrho\right) \right).$$

On the other hand, whenever $\tilde{x}^{(k)} \in \mathcal{L}_\varrho$, the sequence is forced to stay at y_ϱ, and we have the trivial inequality

$$\mathbb{E}\left[\left\| \tilde{x}^{(k+1)} - y_\varrho \right\|^2 \middle| \tilde{x}^{(k)}, w^{(k)} \right] \leq \left\| \tilde{x}^{(k)} - y_\varrho \right\|^2 + 0.$$

If we define $z^{(k)}$ through

$$z^{(k)} = \begin{cases} 2\alpha \left(f\left(\tilde{x}^{(k)}\right) - f\left(y_\varrho\right) \right) - \alpha^2\varphi^2\kappa & \text{if } \tilde{x}^{(k)} \notin \mathcal{L}_\varrho \\ 0 & \text{if } \tilde{x}^{(k)} \in \mathcal{L}_\varrho, \end{cases}$$

we can write

$$\mathbb{E}\left[\left\| \tilde{x}^{(k+1)} - y_\varrho \right\|^2 \middle| \tilde{x}^{(k)}, w^{(k)} \right] \leq \left\| \tilde{x}^{(k)} - y_\varrho \right\|^2 - z^{(k)}, \quad \forall k \geq \theta. \tag{3.12}$$

When $\tilde{x}^{(k)} \notin \mathcal{L}_\varrho$, we have

$$\left(f\left(\tilde{x}^{(k)}\right) - f\left(y_\varrho\right) \right) \geq \frac{1}{\varrho} + \frac{\alpha\varphi^2\kappa}{2},$$

which is equivalent to

$$z^{(k)} = 2\alpha \left(f\left(\tilde{x}^{(k)}\right) - f\left(y_\varrho\right) \right) - \alpha^2\varphi^2\kappa \geq \frac{2\alpha}{\varrho}.$$

If we take the expectation of (3.12), the result is

$$\mathbb{E}\left[\left\|\tilde{x}^{(k+1)}-y_\varrho\right\|^2\right] \leq \mathbb{E}\left[\left\|\tilde{x}^{(k)}-y_\varrho\right\|^2\right] - \mathbb{E}\left[z^{(k)}\right], \quad \forall k \geq \theta,$$

and starting from $x^{(\theta)}$, we recursively get

$$\mathbb{E}\left[\left\|\tilde{x}^{(k+1)}-y_\varrho\right\|^2\right] \leq \mathbb{E}\left[\left\|\tilde{x}^{(\theta)}-y_\varrho\right\|^2\right] - \mathbb{E}\left[\sum_{i=\theta}^{k} z^{(i)}\right], \quad \forall k \geq \theta. \tag{3.13}$$

Furthermore, from the iterations (3.4) and the bounded subgradients assumption, we have that $\left\|\tilde{x}^{(\theta)}-y_\varrho\right\| \leq \left\|\tilde{x}^{(0)}-y_\varrho\right\| + \alpha\varphi\sum_{m=0}^{\theta-1} R_i^{(m)}$. This means that

$$\mathbb{E}\left[\left\|\tilde{x}^{(\theta)}-y_\varrho\right\|^2\right] \leq \left\|\tilde{x}^{(0)}-y_\varrho\right\|^2 + 2\alpha\varphi\theta n\left\|\tilde{x}^{(0)}-y_\varrho\right\| + \omega(\theta,\alpha,\varphi),$$

where $\omega(\theta,\alpha,\varphi) = \mathbb{E}[(\alpha\varphi\sum_{m=0}^{\theta-1} R_i^{(m)})^2]$ is a function that we know is bounded from Lemma 3.3.4. Let $\tilde{\tau}$ be the stopping time defined as

$$\tilde{\tau} = \inf\{t|\tilde{x}^{(t)} \in \mathcal{L}_\varrho,\, t \geq \theta,\, t \in \mathbb{N}\},$$

then $\tilde{\tau}$ is the number of non-zero elements in the non-negative sequence $\{z^{(k)}\}_{k=\theta}^{\infty}$. We have that the limit $\sum_{k=\theta}^{\infty} z^{(k)}$ either exists finitely or is equal to infinity since the partial sums are monotonic. Hence $\sum_{k=\theta}^{\infty} z^{(k)} \geq \frac{2\alpha}{\varrho}\tilde{\tau}$. By letting k go to infinity in (3.13) and using the non-negativity of a norm, we have

$$\begin{aligned} 0 &\leq \left\|\tilde{x}^{(0)}-y_\varrho\right\|^2 + 2\alpha\varphi\theta n\left\|\tilde{x}^{(0)}-y_\varrho\right\| + \omega(\theta,\alpha,\varphi) - \mathbb{E}\left[\sum_{i=\theta}^{\infty} z_i\right] \\ &\leq \left\|\tilde{x}^{(0)}-y_\varrho\right\|^2 + 2\alpha\varphi\theta n\left\|\tilde{x}^{(0)}-y_\varrho\right\| + \omega(\theta,\alpha,\varphi) - \frac{2\alpha}{\varrho}\mathbb{E}\left[\tilde{\tau}\right] \end{aligned}$$

and the bound

$$\mathbb{E}\left[\tilde{\tau}\right] \leq \frac{\varrho}{2\alpha}\left(\left\|x^{(0)}-y_\varrho\right\|^2 + 2\alpha\varphi\theta n\left\|x^{(0)}-y_\varrho\right\| + \omega(\theta,\alpha,\varphi)\right),$$

where we used that $\tilde{x}^{(0)} = x^{(0)}$. Thus, the stopping time $\tilde{\tau}$ is almost surely finite and at least one element in the sequence $\{\hat{x}^{(k)}\}_{k=\theta}^{\infty}$ will be in the set $\mathcal{L}_\varrho$. Since $\{\hat{x}^{(k)}\}_{k=\theta}^{\infty}$ is a subsequence of $\{x^{(l)}\}_{l=\tilde{l}}^{\infty}$ with $\tilde{l} = \sum_{m=0}^{\theta-1} R_i^{(m)} \geq \theta$, it follows that at least one element of $\{x^{(l)}\}_{l=\theta}^{\infty}$ will be in the set $\mathcal{L}_\varrho$. Therefore, we have that

$$\inf_{l\geq\theta} f(x^{(l)}) \leq f\left(y_\varrho\right) + \frac{1}{\varrho} + \frac{\alpha\varphi^2\kappa}{2},$$

and since the choice of θ is arbitrary, we have that

$$\liminf_{l\to\infty} f(x^{(l)}) \leq f\left(y_\varrho\right) + \frac{1}{\varrho} + \frac{\alpha\varphi^2\kappa}{2}.$$

By letting ϱ go to infinity and noting that Lemma 3.3.5 holds for all i, we have shown part a).

Now we proceed with part b), and the proof idea is the same as in part a); we show that the iterates will always enter a special level set. If $\sup_{x\in\mathcal{X}} f(x) \leq f^\star + \frac{\alpha\varphi^2\kappa}{2} + \delta$ or $f(x_0) \leq f^\star + \frac{\alpha\varphi^2\kappa}{2} + \delta$, then the claim in b) is trivially fulfilled. Otherwise, let $\mathcal{L}_\delta$ be the level set defined by

$$\mathcal{L}_\delta = \left\{ x \in \mathcal{X} \middle| f(x) \leq f^\star + \frac{\alpha\varphi^2\kappa}{2} + \delta \right\}.$$

Define the sequence $\{\tilde{x}^{(k)}\}_{k=0}^{\infty}$ as follows

$$\tilde{x}^{(k)} = \begin{cases} \hat{x}^{(k)} & \text{if } \hat{x}_j \notin \mathcal{L}_\delta \ \forall j \leq k \\ \check{x} \in \mathcal{X}^\star & \text{otherwise,} \end{cases}$$

where $\check{x}$ is an arbitrary point in $\mathcal{X}^\star$. When $\tilde{x}^{(k)} \notin \mathcal{L}_\delta$, Lemma 3.3.5 gives us

$$\mathbb{E}\left[\left\| \tilde{x}^{(k+1)} - \check{x} \right\|^2 \middle| \tilde{x}^{(k)}, w^{(k)} \right] \leq \left\| \tilde{x}^{(k)} - \check{x} \right\|^2 + \alpha^2\varphi^2\kappa - 2\alpha\left(f\left(\tilde{x}^{(k)}\right) - f\left(\check{x}\right) \right).$$

Otherwise, when $\tilde{x}^{(k)} \in \mathcal{L}_\delta$, the sequence will stay at $\check{x}$, and we have the trivial inequality

$$\mathbb{E}\left[\left\| \tilde{x}^{(k+1)} - \check{x} \right\|^2 \middle| \tilde{x}^{(k)}, w^{(k)} \right] \leq \left\| \tilde{x}^{(k)} - \check{x} \right\|^2 + 0.$$

By defining $z^{(k)}$ through

$$z^{(k)} = \begin{cases} 2\alpha\left(f\left(\tilde{x}^{(k)}\right) - f\left(\check{x}\right) \right) - \alpha^2\varphi^2\kappa & \text{if } \tilde{x}^{(k)} \notin \mathcal{L}_\delta \\ 0 & \text{if } \tilde{x}^{(k)} \in \mathcal{L}_\delta, \end{cases}$$

we have $z^{(k)} \geq 2\alpha\delta$ if $\tilde{x}^{(k)} \notin \mathcal{L}_\delta$, and we can write

$$\mathbb{E}\left[\left\| \tilde{x}^{(k+1)} - \check{x} \right\|^2 \middle| \tilde{x}^{(k)}, w^{(k)} \right] \leq \left\| \tilde{x}^{(k)} - \check{x} \right\|^2 - z^{(k)}, \quad \forall k. \tag{3.14}$$

Let $\tilde{\tau}$ be the stopping time defined as

$$\tilde{\tau} = \inf\{t | \tilde{x}_t \in \mathcal{L}_\delta,\, t \geq 0,\, t \in \mathbb{N}\},$$

then $\tilde{\tau}$ is the random number of non-zero elements in the non-negative sequence $\{z^{(k)}\}_{k=0}^{\infty}$ and $\sum_{k=0}^{\infty} z^{(k)} \geq 2\alpha\delta\tilde{\tau}$, where the limit $\sum_{k=0}^{\infty} z^{(k)}$ either exists finitely or is equal to infinity, since the partial sums are monotonic. By letting k go to infinity in (3.14) and using the non-negativity of a norm, we have

$$0 \leq \left\|\tilde{x}^{(0)} - \check{x}\right\|^2 - \mathbb{E}\left[\sum_{i=0}^{\infty} z_i\right] \leq \left\|\tilde{x}^{(0)} - \check{x}\right\|^2 - 2\alpha\delta\mathbb{E}\left[\tilde{\tau}\right] \tag{3.15}$$

and the bound

$$\mathbb{E}\left[\tilde{\tau}\right] \leq \frac{1}{2\alpha\delta}\left\|\tilde{x}^{(0)} - \check{x}\right\|^2 = \frac{1}{2\alpha\delta}\left\|x^{(0)} - \check{x}\right\|^2. \tag{3.16}$$

Now let τ be the stopping time defined as

$$\tau = \inf\{t | x^{(t)} \in \mathcal{L}_\delta,\, w^{(t)} = i,\, t \geq 0,\, t \in \mathbb{N}\}.$$

This means that the stopping conditions will be fulfilled when $x^{(t)}$ is in the set $\mathcal{L}_\delta$ and the Markov chain is in state i; note that $f(x^{(\tau)}) \leq f^\star + \frac{\alpha\varphi^2\kappa}{2} + \delta$. By using the recurrence time $R_i^{(k)}$, which counts the number of elements in the original sequence $\{x^{(l)}\}_{l=0}^{\infty}$ between the elements in the sampled sequence $\{\hat{x}^{(k)}\}_{k=0}^{\infty}$, we can write

$$\tau = \sum_{k=1}^{\tilde{\tau}} R_i^{(k-1)},$$

where $\tilde{\tau} \geq 1$ since $x^{(0)} \notin \mathcal{L}_\delta$ by assumption. Since $\tilde{\tau}$ is a stopping time for the sequence $\{\hat{x}^{(0)}, \hat{x}^{(1)}, ...\}$, occurrence of the event $\{\tilde{\tau} \geq j\}$ is decided by the sequence $\{\hat{x}^{(0)}, ..., \hat{x}^{(j-1)}\}$. In particular, $I\{\tilde{\tau} \geq j\} = \prod_{m=0}^{j-1} I\{\hat{x}^{(m)} \notin \mathcal{L}_\delta\}$, where $I\{\cdot\}$ is the indicator function of the event $\{\cdot\}$. Furthermore, due to the construction of $\{\hat{x}^{(k)}\}_{k=0}^{\infty}$ and $\{R_i^{(k)}\}_{k=0}^{\infty}$, and the Markov property of the sequence $\{w^{(k)}\}_{k=0}^{\infty}$, the recurrence times $R_i^{(j-1)}, R_i^{(j)}, R_i^{(j+1)}, ...$ are independent of $\hat{x}^{(j-1)}, \hat{x}^{(j-2)}, \hat{x}^{(j-3)}, ...$ More specifically, we have that

$$\mathbb{E}\left[I\{\tilde{\tau} \geq j\}R_i^{(j-1)}\right] = \mathbb{E}\left[\prod_{m=0}^{j-1} I\{\hat{x}^{(m)} \notin \mathcal{L}_\delta\}R_i^{(j-1)}\right] = \mathbb{P}\{\tilde{\tau} \geq j\}\mathbb{E}\left[R_i^{(j-1)}\right],$$

where $\mathbb{P}\{\cdot\}$ denotes the probability of the event $\{\cdot\}$. Using the previous properties with a Wald's identity (see, e.g., Chung (1974, Theorem 5.5.3)) type of argument,

we have

$$\mathbb{E}[\tau] = \sum_{l=1}^{\infty} \mathbb{E}\left[I\{\tilde{\tau} = l\} \sum_{k=1}^{\tilde{\tau}} R_i^{(k-1)} \right] = \tag{3.17a}$$

$$= \sum_{l=1}^{\infty} \sum_{k=1}^{l} \mathbb{E}\left[I\{\tilde{\tau} = l\} R_i^{(k-1)} \right] = \sum_{k=1}^{\infty} \sum_{l=k}^{\infty} \mathbb{E}\left[I\{\tilde{\tau} = l\} R_i^{(k-1)} \right] = \tag{3.17b}$$

$$= \sum_{k=1}^{\infty} \mathbb{E}\big[I\{\tilde{\tau} \geq k\} R_i^{(k-1)} \big] = \sum_{k=1}^{\infty} \mathbb{P}\{\tilde{\tau} \geq k\} \mathbb{E}\left[R_i^{(k-1)} \right] = \mathbb{E}[\tilde{\tau}] \mathbb{E}\big[R_i^{(0)} \big] \leq \tag{3.17c}$$

$$\leq \frac{n}{2\alpha\delta} \left\| x^{(0)} - \check{x} \right\|^2. \tag{3.17d}$$

The change of summation order in (3.17b) holds since the series converges absolutely

$$\sum_{k=1}^{\infty} \sum_{l=k}^{\infty} \left| \mathbb{E}\left[I\{\tilde{\tau} = l\} R_i^{(k-1)} \right] \right| = \sum_{k=1}^{\infty} \sum_{l=k}^{\infty} \mathbb{E}\left[I\{\tilde{\tau} = l\} R_i^{(k-1)} \right] = \mathbb{E}[\tilde{\tau}] \mathbb{E}\left[R_i^{(0)} \right] < \infty,$$

where we used the non-negativity of $\tilde{\tau}$ and $R_i^{(k)}$. The relation $\sum_{k=1}^{\infty} \mathbb{P}\{\tilde{\tau} \geq k\} = \mathbb{E}[\tilde{\tau}]$, used in (3.17c), follows from Chung (1974, Theorem 3.2.1). Since (3.17d) holds for arbitrary $\check{x}$ in $\mathcal{X}^\star$, we can replace $\|x^{(0)} - \check{x}\|^2$ with $\big(\mathrm{dist}_{\mathcal{X}^\star}(x^{(0)})\big)^2$. □

Remark 3.3.7. When the n-node network is connected and Assumption 3.2.1 holds, Lemma 2.3.5 and Theorem 3.3.6 imply that MC can be devised using only local topology information and that the iterations (3.4) can be executed using only peer-to-peer communication and that they converge in the sense of Theorem 3.3.6.

3.3.3 Comparison with Existing Incremental Subgradient Algorithms

For the DISM and the RISM, there exist results of the same type as Theorem 3.3.6, i.e.,

$$\min_{0 \leq l \leq \tau} f(x^{(l)}) = f^\star + \alpha\beta + \nu \text{ with } \mathbb{E}[\tau] \leq \frac{\rho}{\alpha\nu}, \tag{3.18}$$

where β and ρ are positive constants that depend on the algorithm (α is still the constant stepsize). To compare the algorithms, we need the minimum expected number of iterations needed for each algorithm for a given accuracy ($\min_{0 \leq l \leq \tau} f(x^{(l)}) = f^\star + \gamma$). For the general case (3.18), we get the following optimization problem

$$\begin{array}{lll} \underset{\alpha,\nu}{\text{minimize}} & \frac{\rho}{\alpha\nu} & \\ \text{subject to} & \alpha\beta + \nu \leq \gamma & \Leftrightarrow \\ & \alpha \geq 0,\ \nu \geq 0 & \end{array} \quad \begin{array}{lll} \underset{\alpha,\nu}{\text{maximize}} & \alpha\nu & \\ \text{subject to} & \alpha\beta + \nu = \gamma & \Rightarrow \\ & \alpha \geq 0,\ \nu \geq 0 & \end{array} \quad \begin{cases} \alpha^\star = \frac{\gamma}{2\beta} \\ \nu^\star = \frac{\gamma}{2}. \end{cases}$$

Table 3.1: Expected number of iterations, $\mathbb{E}[\tau]$, needed to reach the accuracy $\min_{0\leq l\leq\tau} f(x^{(l)}) \leq f^\star + \gamma$. For brevity[2], let $\vartheta = n\varphi^2\gamma^{-2}\left(\operatorname{dist}_{\mathcal{X}^\star}(x_0)\right)^2$.

Algorithm	$\mathbb{E}[\tau]$
DISM	$n^2\vartheta$
RISM	$n\vartheta$
MISM	$\kappa\vartheta$

Using these optimal values, we compute an upper bound of the expected number of iterations, $\mathbb{E}[\tau]$, needed to reach the accuracy $\min_{0\leq l\leq\tau} f(x^{(l)}) \leq f^\star + \gamma$ for the DISM, RISM, and MISM. The results are presented in Table 3.1. Since

$$\kappa \geq \mathbb{E}\left[\left(R_i^{(k)}\right)^2\right] = \mathbb{E}\left[\left(\sum_{i=1}^n v_i^{(k)}\right)^2\right] \geq \mathbb{E}\left[\sum_{i=1}^n \left(v_i^{(k)}\right)^2\right] \geq \mathbb{E}\left[\sum_{i=1}^n v_i^{(k)}\right] = n,$$

where we used the non-negativity and integrality of $v_i^{(k)}$, the results in Table 3.1 indicate that the RISM is the best algorithm, and that the ranking between the DISM and the MISM will depend on the topology of the network as well as the transition probability matrix of the Markov chain. However, it is not only the expected number of iterations that are of interest, and in applications, the ease of implementation and energy consumption are crucial. Experiments show that the MISM has favorable properties in these two respects, as reported in Johansson et al. (2007), and we address implementation issues and performance in Chapter 6.

It is interesting to note that we can recover the DISM and RISM from the MISM by choosing the transition probability matrix in the following way:

$$P_{\text{DISM}} = \begin{pmatrix} 0 & 1 & 0 & \dots \\ 0 & 0 & 1 & \dots \\ \vdots & & & \vdots \\ 1 & 0 & 0 & \dots \end{pmatrix} \text{ and } P_{\text{RISM}} = \begin{pmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & \ddots & \vdots \\ \frac{1}{n} & \cdots & \frac{1}{n} \end{pmatrix}$$

with

$$\mathbb{E}[R_{\text{DISM}}] = n^2 \text{ and } \mathbb{E}[R_{\text{RISM}}] = 2n^2 - n.$$

The transition matrix P_{DISM} will make the Markov chain deterministically explore the topology in a logical ring and $R_i^{(k)} = n$. Note that the Markov chain corresponding to P_{DISM} does not satisfy Assumption 3.3.1, but the analysis in Theorem 3.3.6

[2]The constant φ is defined in a slightly different way for DISM and RISM in Nedić and Bertsekas (2001). There it is assumed that the norm of the subgradients for the actual trajectory of the algorithms are upper bounded by φ. This is more general and less conservative than our definition of φ, but it is very hard to check if it is fulfilled and therefore not practical. Our analysis holds for the less conservative definition of φ as well.

still holds. The transition matrix P_{RISM} will make the Markov chain jump to any node in the topology with equal probability at each time step, precisely as the RISM, and $\mathbb{E}[R_{\mathrm{RISM}}] = 2n^2 - n$ is given by Lemma 3.3.4. The convergence bound given by the MISM analysis for P_{DISM} is identical with the convergence bound given by the DISM analysis in Nedić and Bertsekas (2001). On the other hand, the convergence bound given by the MISM analysis for P_{RISM} is much worse than the original RISM result. This is due to the fact that in the original RISM analysis, all iterates are analyzed, while in the MISM analysis, only iterates in the starting state, which is arbitrary, are analyzed.

3.4 Distributed Subgradient Method using Consensus Iterations

As mentioned in Section 3.3, subgradient methods are popular for solving problems of the type (3.1). The subgradient method can be implemented in an incremental fashion as proposed in Bertsekas et al. (2003). This entails changing the variable $x^{(k)}$ incrementally through n steps, in each iteration using only the subgradient corresponding to a single component function f_i. The advantage of this method from a computational aspect is that it can be performed in a distributed way by assigning each component function to a processing unit or "agent", which performs the local incremental update on the variable x. This means that $x^{(k)}$ needs to be passed between the agents, which perform a subgradient update using only a single subgradient corresponding to the agent's component function. This incremental subgradient scheme has advantages over the standard one in terms of convergence rate and distribution of computations. However, in its usual form the incremental updates are performed in a sequential manner, which assumes that the variable $x^{(k)}$ passes through all agents either in a cyclic or randomized sequence. Implementation of such a communication scheme can sometimes be problematic in practice. This predicament was addressed in Section 3.3, where we developed a new randomized algorithm.

Notice that we can think of the computing agents mentioned above as each having a copy of the decision variable x, which they maintain locally and update based on the value obtained from the previous subiteration (the preceding agent in the update sequence) and the local subgradient of the component function evaluated at this value. Under appropriate assumptions and using a properly chosen diminishing stepsize, the subgradient iterations converge asymptotically to an optimizer $x^\star$ of the problem. This means that eventually all "local" versions of the decision variable converge to the same value. This resembles to some extent agreement or consensus problems in multi-agent systems, which has a long history and has received renewed interest in the recent literature (Olfati-Saber et al., 2007). It is thus interesting to investigate whether the convergence properties of certain consensus algorithms can be combined with subgradient iterations in order to optimize problems of type (3.1) using a wider variety of communication topologies than what the

standard incremental subgradient method allows.

3.4.1 Algorithm

Inspired by Nedić and Ozdaglar (2007*b*,*a*), we propose to combine the subgradient iterations with a number of consensus iterations in the following way:

$$x_i^{(k+1)} = \mathcal{P}_{\mathcal{X}}\left[\sum_{j=1}^{n}[W^{\theta}]_{ij}\left(x_j^{(k)} - \alpha^{(k)}a_j(x_j^{(k)})\right)\right], \tag{3.19}$$

where $a_j(x_j^{(k)}) \in \partial f_j(x_j^{(k)})$ and $[W^{\theta}]_{ij}$ denotes the element of W^{θ} in the ith row and jth column. We call this algorithm the consensus subgradient method, CSM. Agents maintain their local variable $x_i^{(k)}$ and perform the update procedure of (3.19) in parallel. First, they exchange information with neighboring agents for θ number of consensus iterations. More specifically, (3.19) implies that each agent i runs θ number of consensus iterations with its neighbors, defined by (2.7), for each row in the local vector $x_i^{(k)}$. Then, all agents implement the component subgradient update locally. The total number of iterations in the algorithm up to step k is therefore $k\theta$. For a more compact notation we define

$$u_i^{(k)} = x_i^{(k)} - \alpha^{(k)}a_i(x_i^{(k)}) \tag{3.20}$$

and

$$v_i^{(k)} = \sum_{j=1}^{n}[W^{\theta}]_{ij}u_j^{(k)},\ i = 1, ..., n. \tag{3.21}$$

Let us assume for the moment that in each subgradient iteration $\theta \to \infty$, and the consensus updates converge to the average of the initial values (this holds if W fulfills Assumption 2.3.2). Then, for all $i = 1, \ldots, n$ and $x_i^{(0)} \in \mathbb{R}^{\xi}$,

$$\lim_{\theta\to\infty}\sum_{j=1}^{n}\left([W^{\theta}]_{ij}u_j^{(0)}\right) = \frac{1}{n}\sum_{j=1}^{n}\left(x_j^{(0)} - \alpha^{(0)}a_j(x_j^{(0)})\right).$$

Let us denote the initial state of the projected consensus variable with

$$\breve{x}^{(1)} = \mathcal{P}_{\mathcal{X}}\left[\frac{1}{n}\sum_{j=1}^{n}\left(x_j^{(0)} - \alpha^{(0)}a_j(x_j^{(0)})\right)\right].$$

In the next iteration, each agent will possess the same value $\breve{x}_1$, thus the local subgradients a_j will be evaluated at the same point. The update procedure in (3.19) for $k \geq 1$ will thus be equivalent to

$$\breve{x}^{(k+1)} = \mathcal{P}_{\mathcal{X}}\left[\frac{1}{n}\sum_{j=1}^{n}\left(\breve{x}^{(k)} - \alpha^{(k)}a_j(\breve{x}^{(k)})\right)\right] = \mathcal{P}_{\mathcal{X}}\left[\breve{x}^{(k)} - \alpha^{(k)}\frac{1}{n}\sum_{j=1}^{n}(a_j(\breve{x}^{(k)}))\right]. \tag{3.22}$$

This is the same process as the standard subgradient method of (3.2) performed on the consensus variable $\breve{x}^{(k)}$ with a stepsize of $\alpha^{(k)}/n$. Convergence analysis of this scheme can be done following the procedure in for example Kiwiel (2004) or Bertsekas et al. (2003).

We are interested however in a more realistic scenario, where the consensus iterations are performed only for a finite number of θ steps. We intend to analyze such a scheme with the use of properly chosen approximate subgradients and the approximation error of the average consensus process achieved in a finite number of steps. Finally, we assume that the stepsize in (3.19) is constant, $\alpha^{(k)} = \alpha$, which makes the algorithm easily implementable.

Our algorithm is similar to the algorithm proposed in Nedić and Ozdaglar (2007*b*,*a*), but there are some differences. Firstly, Nedić and Ozdaglar (2007*b*,*a*) propose to use one consensus step per subgradient iteration, which in our notation can be written as

$$x_i^{(k+1)} = \left(\sum_{j=1}^{n} [W^{(k)}]_{ij} x_j^{(k)}\right) - \alpha_i^{(k)} a_i(x_i^{(k)})$$

where $W^{(k)}$ is time varying. Whereas (3.19) allows us to use more consensus iterations to make the step taken more similar at each node. Secondly, they use the partially asynchronous model stemming from the seminal work of Tsitsiklis (1984), which is much more general than our synchronous model. Using the average of the nodes' iterates, $\bar{x}^{(k)} = \sum_{i=1}^{n} x_i^{(k)}/n$, they show that the individual nodes iterates are close to this average under some communication assumptions, where connectivity in the long run and bounded intercommunication intervals are the most crucial ones. Furthermore, they show how that the time average of the nodes' iterates, $\hat{x}_i^{(k)} = \sum_{j=1}^{k} x_i^{(j)}/k$, as well as the time average of the average of the nodes' iterates, $\hat{\bar{x}}^{(k)} = \sum_{j=1}^{k} \bar{x}_i^{(j)}/k$, asymptotically converge to something close to the optimal set. As we will see later, we also use the approach of analyzing $\bar{x}^{(k)}$ and provide bounds on the distance between $\bar{x}^{(k)}$ and $x_i^{(k)}$. However, we provide a convergence analysis of $\bar{x}^{(k)}$ and not of $\hat{\bar{x}}^{(k)}$, and most importantly, we also include projections on the closed and convex set $\mathcal{X}$ in the analysis, which allow us to handle constrained optimization problems.

The following three lemmas will be instrumental in the convergence analysis of the proposed scheme. We will denote the average value of the local variables at time k with $\bar{x}^{(k)} = \frac{1}{n}\sum_{i=1}^{n} x_i^{(k)}$ and $\bar{v}^{(k)} = \frac{1}{n}\sum_{i=1}^{n} v_i^{(k)}$.

The following inequalities serve to characterize the Euclidean distance of agent variables from each other and from their average.

Lemma 3.4.1. *1) If* $\left\|x_i^{(k)} - x_j^{(k)}\right\| \leq \beta$ *for all* $i, j = 1, ..., n$, *then*

$$\left\|x_j^{(k)} - \bar{x}^{(k)}\right\| = \left\|x_j^{(k)} - \frac{1}{n}\sum_{i=1}^{n} x_i^{(k)}\right\| \leq \frac{n-1}{n}\beta.$$

2) If $\left\|x_i^{(k)} - \bar{x}^{(k)}\right\| \leq \beta$ *for all* $i = 1, ..., n$, *then* $\left\|x_i^{(k)} - x_j^{(k)}\right\| \leq 2\beta$.

Proof. 1)

$$\left\|x_j^{(k)} - \bar{x}^{(k)}\right\| = \frac{1}{n}\left\|nx_j^{(k)} - \sum_{i=1}^{n} x_i^{(k)}\right\| \leq \frac{1}{n}\left(\sum_{i=1, i\neq j}^{n}\left\|x_i^{(k)} - x_j^{(k)}\right\|\right) \leq \frac{n-1}{n}\beta$$

2)

$$\left\|x_i^{(k)} - x_j^{(k)}\right\| \leq \left\|x_i^{(k)} - \bar{x}\right\| + \left\|x_j^{(k)} - \bar{x}\right\| \leq 2\beta$$

□

Lemma 3.4.2. *If* $\left\|[y^{(k)}]_i - [y^{(k)}]_j\right\| \leq \sigma$ *for all* $i, j = 1, ..., n$ *and* $y^{(k+1)} = W^\theta y^{(k)}$, *with* W *fulfilling Assumption 2.3.2, then* $\left\|[y^{(k+1)}]_i - [y^{(k+1)}]_j\right\| \leq 2\gamma^\theta n\sigma$ *for all* $i, j = 1, ..., n$.

Proof. Let us write $y^{(k)}$ as $y^{(k)} = \bar{y}^{(k)} + a^{(k)}$ with $\sum_{i=1}^{n}[a^{(k)}]_i = 0$ and $\bar{y}^{(k)} = \mathbf{1}_n\mathbf{1}_n^\intercal y^{(k)}/n$. The results of Lemma 3.4.1 show that $\left\|[a^{(k)}]_i\right\| \leq \sigma$ for all i. Furthermore, $y^{(k+1)} = W^\theta(\bar{y}^{(k)} + a^{(k)}) = \bar{y}^{(k)} + W^\theta(a^{(k)} - \mathbf{0}_n)$. Now we have,

$$\left\|[y^{(k+1)} - \bar{y}^{(k)}]_i\right\| \leq \left\|y^{(k+1)} - \bar{y}^{(k)}\right\| = \left\|W^\theta(a^{(k)} - \mathbf{0}_n)\right\| = \left\|\left(W^\theta - \frac{\mathbf{1}_n\mathbf{1}_n^\intercal}{n}\right)a^{(k)}\right\|$$

$$\leq \left\|W^\theta - \left(\frac{\mathbf{1}_n\mathbf{1}_n^\intercal}{n}\right)^\theta\right\|\left\|a^{(k)}\right\| \leq \left\|W - \frac{\mathbf{1}_n\mathbf{1}_n^\intercal}{n}\right\|^\theta\left\|a^{(k)}\right\| \leq \gamma^\theta\left\|a^{(k)}\right\| \leq \gamma^\theta n\sigma,$$

where we used that $\left\|W - \frac{\mathbf{1}_n\mathbf{1}_n^\intercal}{n}\right\| = \rho\left(W - \frac{\mathbf{1}_n\mathbf{1}_n^\intercal}{n}\right)$, which holds for symmetric matrices. Finally, from the above discussion and Lemma 3.4.1, we have

$$\left\|[y^{(k+1)}]_i - [y^{(k+1)}]_j\right\| \leq 2\gamma^\theta n\sigma \text{ for all } i, j = 1, ..., n.$$

□

The following lemma establishes a lower bound on the number of consensus steps, which will ensure that the local variables will remain in a ball of radius β of their average, from one iteration to the next.

Lemma 3.4.3. *Let* $\{x_1^{(k)}, ..., x_n^{(k)}\}_{k=0}^{\infty}$ *be generated by (3.19) under Assumptions 3.2.1 and 2.3.2. If* $\left\|v_i^{(k)} - \bar{v}^{(k)}\right\| \leq \beta$ *for all* i *and*

$$\theta \geq \frac{\log(\beta) - \log(4\xi n(\beta + \alpha\varphi))}{\log(\gamma)},$$

then $\left\|v_i^{(k+1)} - \bar{v}^{(k+1)}\right\| \leq \beta$ *for all* i.

Proof. Using Lemma 3.4.1, the closeness condition means that $\left\|v_i^{(k)} - v_j^{(k)}\right\| \leq 2\beta$ for all $i, j = 1, ..., n$, which implies that $\left\|\left[v_i^{(k)} - v_j^{(k)}\right]_l\right\| \leq 2\beta$ for all $i, j = 1, ..., n$ and $l = 1, ..., m$.

We can now decide the distance between the iterates before the consensus step.

$$\begin{aligned}\left\|\left[u_i^{(k+1)}\right]_l - \left[u_j^{(k+1)}\right]_l\right\| &= \left\|\left[\mathcal{P}_{\mathcal{X}}[v_i^{(k)}] - \alpha a_i(\mathcal{P}_{\mathcal{X}}[v_i^{(k)}]) - \mathcal{P}_{\mathcal{X}}[v_j^{(k)}] + \alpha a_j(\mathcal{P}_{\mathcal{X}}[v_j^{(k)}])\right]_l\right\| \\ &\leq \left\|v_i^{(k)} - v_j^{(k)}\right\| + 2\alpha\varphi \leq 2(\beta + \alpha\varphi), \end{aligned} \tag{3.23}$$

where we used the non-expansive property of the projection on a convex set. Furthermore, we have

$$\begin{aligned}\left\|v_i^{(k+1)} - v_j^{(k+1)}\right\| &= \left\|\sum_{l=1}^{n}[W^\theta]_{il}u_l^{(k+1)} - \sum_{l=1}^{n}[W^\theta]_{jl}u_l^{(k+1)}\right\| \\ &\leq \sum_{o=1}^{\xi}\left\|\left[\sum_{l=1}^{n}[W^\theta]_{il}u_l^{(k+1)} - \sum_{l=1}^{n}[W^\theta]_{jl}u_l^{(k+1)}\right]_o\right\|.\end{aligned}$$

Consider now one of the terms in the sum above and notice that

$$\begin{aligned}&\left\|\left[\sum_{l=1}^{n}[W^\theta]_{il}u_l^{(k+1)} - \sum_{l=1}^{n}[W^\theta]_{jl}u_l^{(k+1)}\right]_o\right\| \\ &= \left\|\sum_{l=1}^{n}[W^\theta]_{il}\left[u_l^{(k+1)}\right]_o - \sum_{l=1}^{n}[W^\theta]_{jl}\left[u_l^{(k+1)}\right]_o\right\| = \left\|\left[W^\theta y^{(k)}\right]_i - \left[W^\theta y^{(k)}\right]_j\right\|,\end{aligned}$$

with $y^{(k)} = \left(\left[u_1^{(k+1)}\right]_o \ \ ... \ \ \left[u_n^{(k+1)}\right]_o\right)^\intercal$. Using Lemma 3.4.2 and (3.4.1), which states that $\left\|\left[y^{(k)}\right]_i - \left[y^{(k)}\right]_j\right\| \leq 2(\beta + \alpha\varphi)$, we obtain $\left\|\left[W^\theta y^{(k)}\right]_i - \left[W^\theta y^{(k)}\right]_j\right\| \leq 4\gamma^\theta n(\beta + \alpha\varphi)$.

Combining the above results yields $\left\|v_i^{(k+1)} - v_j^{(k+1)}\right\| \leq 4\gamma^\theta\xi n(\beta + \alpha\varphi)$, and $\left\|v_i^{(k+1)} - v_j^{(k+1)}\right\| \leq \beta$ is fulfilled if θ is set to

$$\theta \geq \frac{\log(\beta) - \log(4\xi n(\beta + \alpha\varphi))}{\log(\gamma)}.$$

□

3.4.2 Convergence Analysis

We will prove convergence of the algorithm in two cases: Firstly, convergence is proved when the feasible set is the space $\mathbb{R}^\xi$, i.e., the unconstrained optimization problem. Secondly, with an additional assumption on the objective functions, convergence is proved for a general convex feasible set.

Unconstrained Case

In the following, we will analyze the unconstrained case and we make the following assumption.

Assumption 3.4.4. *The feasible set of* (3.1) *is* $\mathcal{X} = \mathbb{R}^{\xi}$.

We need the following Lemma which allows us to interpret the algorithm as an ε-subgradient algorithm.

Lemma 3.4.5. *Under Assumptions 3.2.1 and 2.3.2 and if* $\|x_i^{(k)} - \bar{x}^{(k)}\| \leq \beta$ *for all* $i = 1, ..., n$, *then* $a_i(x_i^{(k)}) \in \partial_{\varepsilon} f_i(\bar{x}^{(k)})$ *and* $\sum_{i=1}^{n} a_i(x_i^{(k)}) \in \partial_{n\varepsilon} f(\bar{x}^{(k)})$, *with* $\varepsilon = 2\beta\varphi$.

Proof. Using the definition (2.1) and the bound on the subgradient in Assumption 3.2.1, leads to

$$\begin{aligned} f_i(x_i^{(k)}) &\geq f_i(\bar{x}^{(k)}) + a_i(\bar{x}^{(k)})^{\intercal}(x_i^{(k)} - \bar{x}^{(k)}) \geq f_i(\bar{x}^{(k)}) - \|a_i(\bar{x}^{(k)})\| \|x_i^{(k)} - \bar{x}^{(k)}\| \\ &\geq f_i(\bar{x}^{(k)}) - \varphi\beta. \end{aligned}$$

For any $y \in \mathcal{X}$, using the subgradient inequality leads to

$$\begin{aligned} f_i(y) &\geq f_i(x_i^{(k)}) + a_i(x_i^{(k)})^{\intercal}(y - x_i^{(k)}) \geq f_i(\bar{x}^{(k)}) + a_i(x_i^{(k)})^{\intercal}(y - x_i^{(k)}) - \varphi\beta \\ &\geq f_i(\bar{x}^{(k)}) + a_i(x_i^{(k)})^{\intercal}(y - \bar{x}^{(k)} + \bar{x}^{(k)} - x_i^{(k)}) - \varphi\beta \\ &\geq f_i(\bar{x}^{(k)}) + a_i(x_i^{(k)})^{\intercal}(y - \bar{x}^{(k)}) - 2\varphi\beta. \end{aligned}$$

Using the definition of an ε-subdifferential (2.4), this implies $a_i(x_i^{(k)}) \in \partial_{2\beta\varphi} f_i(\bar{x}^{(k)})$. Summation of terms yields

$$f(y) \geq f(\bar{x}^{(k)}) + \left(\sum_{i=1}^{n} a_i(x_i^{(k)})\right)^{\intercal}(y - \bar{x}^{(k)}) - n2\beta\varphi.$$

Based on Definition 2.1.5, this implies $\sum_{i=1}^{n} a_i(x_i^{(k)}) \in \partial_{n2\beta\varphi} f(\bar{x}^{(k)})$. □

Now we are ready for the convergence theorem for the unconstrained case.

Theorem 3.4.6. *Under Assumptions 3.2.1, 2.3.2, and 3.4.4, with the sequence* $\{x_1^{(k)}, ..., x_n^{(k)}\}_{k=0}^{\infty}$ *generated by* (3.19) *with*

$$\theta \geq \frac{\log(\beta) - \log(4n\xi(\beta + \alpha\varphi))}{\log(\gamma)}$$

and $\|x_i^{(0)} - \bar{x}^{(0)}\| \leq \beta$, *we have:*
If $f^{\star} = -\infty$, *then*

$$\liminf_{k \to \infty} f(x_i^{(k)}) = -\infty, \; \forall i = 1, ..., n.$$

If $f^\star > -\infty$*, then*

$$\liminf_{k\to\infty} f(x_i^{(k)}) \leq f^\star + \alpha n\varphi^2/2 + 3n\varphi\beta,\ \forall i = 1, ..., n.$$

Proof. From Lemma 3.4.3, we know that $\left\|x_i^{(k)} - \bar{x}^{(k)}\right\| \leq \beta$ for all $i = 1, .., n$ and all $k \geq 0$, since $x_i^{(k)} = v_i^{(k)}$. Furthermore, from Lemma 3.4.5, we know that $\sum_{i=1}^n a_i(x_i^{(k)}) \in \partial_{2n\beta\varphi} f(\bar{x}^{(k)})$ for all $k \geq 0$. Hence, from the definitions of $\bar{x}^{(k)}$ and $x_i^{(k)}$ in combination with the results above, we have

$$\bar{x}^{(k+1)} = \frac{1}{n}\sum_{i=1}^n \left(\sum_{j=1}^n [W^\theta]_{ij}\left(x_j^{(k)} - \alpha a_j(x_j^{(k)})\right)\right) =$$

$$= \frac{1}{n}\sum_{j=1}^n \left(x_j^{(k)} - \alpha a_j(x_j^{(k)})\right) = \bar{x}^{(k)} + \frac{\alpha}{n} h(\bar{x}^{(k)}),$$

with $h(\bar{x}^{(k)}) \in \partial_{2n\beta\varphi} f(\bar{x}^{(k)})$ and $\left\|h(\bar{x}^{(k)})\right\| \leq n\varphi$. This is precisely the approximate subgradient iteration and from Nedić (2002, Proposition 4.1) we have

$$\liminf_{k\to\infty} f(\bar{x}^{(k)}) = -\infty,\ \text{if } f^\star = -\infty$$

and

$$\liminf_{k\to\infty} f(\bar{x}^{(k)}) \leq f^\star + \alpha(n\varphi)^2/(2n) + 2n\varphi\beta,\ \text{if } f^\star > -\infty.$$

By noting that

$$f(x_i^{(k)}) \leq f(\bar{x}^{(k)}) + n\varphi\beta,\ \forall i = 1, ..., n, k \geq 0,$$

we have the desired result. □

Remark 3.4.7. We can get $\liminf_{k\to\infty} f(x_i^{(k)})$ to be arbitrarily close to $f^\star$, by choosing the constants α and β arbitrarily small. Note that the number of required consensus negotiations, θ, to reach a fixed β does not depend on k.

Constrained Case

To show convergence in the constrained case, we need the following additional assumption on the functions f_i.

Assumption 3.4.8. *There exist* $\zeta > 0$ *and* $\tau > 0$ *such that for all* $x \in \mathcal{X}$*,* $a_i(x) \in \partial f_i(x)$*, and* $\nu \in \mathbb{R}^\xi$ *with* $\|\nu\| \leq \tau$*, the following holds:*

$$a_i(x) + \nu \in \partial_\zeta f_i(x).$$

As mentioned before, Assumption 3.4.8 is an additional assumption compared to what is needed for the unconstrained case. However, if the set $\mathcal{X}$ is compact, we can always fix this problem, as the following lemma shows.

Lemma 3.4.9. *Let $\mathcal{X}$ be a closed, convex, and compact set such that $x \in \mathcal{X} \Rightarrow \|x\| \leq \eta$. Furthermore, let $f_i : \mathbb{R}^\xi \to \mathbb{R}$ be a convex function. Then there exists a convex function $\check{f}_i : \mathbb{R}^\xi \to \mathbb{R}$ such that for all $x \in \mathcal{X}$*

$$\nu \in \mathbb{R}^\xi,\ \|\nu\| \leq \tau \text{ and } a_i(x) \in \partial f_i(x) \quad \Rightarrow \quad a_i(x) + \nu \in \partial_\zeta \check{f}_i(x),$$

with $2\eta\tau \leq \zeta$. Furthermore, $\check{f}_i(x) = f_i(x)$ when $x \in \mathcal{X}$.

Proof. Pick any $x \in \mathcal{X}$ and let $a_i(x) \in \partial f_i(x)$, then, due to the subgradient inequality,

$$f_i(z) \geq f_i(x) + (a_i(x) + \nu)^\intercal (z - x) - \zeta, \quad \forall z \in \mathcal{X}$$

is equivalent with

$$0 \geq \nu^\intercal (z - x) - \zeta \geq \nu^\intercal (z - x) - 2\eta\tau, \quad \forall z \in \mathcal{X},$$

which holds since $\nu^\intercal (z-x) \leq \|\nu\|\|z-x\| \leq 2\eta\tau$ by assumption. Furthermore, define $\check{f}_i(z)$ as

$$\check{f}_i(z) = \max\{f_i(z), \varrho_i(z)\}$$

with

$$\varrho_i(z) = \max_{x, a_i, \nu} \big\{ f_i(x) + (a_i(x) + \nu)^\intercal (z - x) - \zeta \,\big|\, x \in \mathcal{X},\ a_i(x) \in \partial f_i(x),\ \|\nu\| \leq \tau \big\},$$

where the maximum in the definition of ϱ_i is attained since the feasible set is compact (in particular, the subdifferential is compact by Rockafellar (1970, Theorem 23.4)). Since $\check{f}_i$ is the pointwise maximum of a family of convex functions, it is convex. In addition, by construction, we have that $f_i(z) \geq \varrho_i(z)$ when $z \in \mathcal{X}$, which implies that $\check{f}_i(z) = f_i(z)$ when $z \in \mathcal{X}$. Finally, also by construction, for any $x \in \mathcal{X}$, $a_i(x) \in \partial f_i(x)$, and $\nu \in \mathbb{R}^\xi$ with $\|\nu\| \leq \tau$, we have that

$$\check{f}_i(z) \geq f_i(x) + (a_i(x) + \nu)^\intercal (z - x) - \zeta, \quad \forall z \in \mathbb{R}^\xi,$$

which implies that $a_i(x) + \nu \in \partial_\zeta \check{f}_i(x)$. □

Since $\check{f}_i(x) = f_i(x)$ when $x \in \mathcal{X}$, we can optimize over $\check{f}_i$ in (3.1) instead of over f_i. It is not needed to actually construct $\check{f}_i$, since it is only used for the convergence proof. Furthermore, some functions directly fulfill Assumption 3.4.8, which obviate the need for $\check{f}_i$ in those cases, as shown in the following example.

Example 3.4.1. Consider $f_i(x) = \psi x^\intercal x$, then Assumption 3.4.8 is fulfilled if $\tau \leq 2\sqrt{\zeta\psi}$ (without $\mathcal{X}$ necessarily being a compact set). This can be shown as follows. Since f_i is differentiable, we have $a_i(x) = 2\psi x$ and

$$\begin{aligned} &f_i(w) - (f_i(x) + (2\psi x + \nu)^\intercal (w - x)) \\ &\quad = \psi(w^\intercal w + x^\intercal x - 2w^\intercal x) + \nu^\intercal (x - w) = \psi(w - x)^\intercal (w - x) + \nu^\intercal (x - w) \\ &\quad \geq \psi \|w - x\|^2 - \|\nu\| \|w - x\| \geq \psi \|w - x\|^2 - \tau \|w - x\| \geq -\tau^2/(4\psi). \end{aligned}$$

Now let $\zeta \geq \tau^2/(4\psi)$ to fulfill Assumption 3.4.8 and we have $\tau \leq 2\sqrt{\zeta\psi}$.

In our further developments we need to keep track of the difference $\bar{x}^{(k)} - \mathcal{P}_{\mathcal{X}}\left[\bar{v}^{(k-1)}\right]$, and to this end, we define $y^{(k)}$ and $z^{(k)}$ as

$$y^{(k)} = \mathcal{P}_{\mathcal{X}}\left[\bar{v}^{(k-1)}\right] \text{ and } z^{(k)} = \bar{x}^{(k)} - y^{(k)}. \tag{3.27}$$

Furthermore, we need the following Lemma, which is similar to Lemma 3.4.5.

Lemma 3.4.10. *Under Assumptions 3.2.1, 2.3.2, and 3.4.8 and if*

$$\left\|v_i^{(k-1)} - \bar{v}^{(k-1)}\right\| \leq \beta$$

for all i and $\nu \in \mathbb{R}^\xi$ with $\|\nu\| \leq \tau$, then

$$(a_i(x_i^{(k)}) + \nu) \in \partial_\varepsilon f_i(y^{(k)})$$

and

$$\sum_{i=1}^{n}(a_i(x_i^{(k)}) + \nu) \in \partial_{n\varepsilon} f_i(y^{(k)})$$

with $\varepsilon = \beta(6\varphi + 3\tau) + \zeta$.

Proof. The proof relies on the iterates being close to each other. Using the assumptions and the non-expansive property of projection on a convex set we can bound the distance $\left\|z^{(k)}\right\|$ as follows

$$\left\|z^{(k)}\right\| = \frac{1}{n}\left\|\sum_{i=1}^{n}\left(\mathcal{P}_{\mathcal{X}}\left[v_i^{(k-1)}\right] - \mathcal{P}_{\mathcal{X}}\left[\bar{v}^{(k-1)}\right]\right)\right\| \leq \frac{1}{n}\sum_{i=1}^{n}\left\|v_i^{(k-1)} - \bar{v}^{(k-1)}\right\| \leq \beta.$$

Using the definition (2.1), the bound on the subgradient in Assumption 3.2.1, the bound above, and Lemma 3.4.1 we obtain

$$\begin{aligned} f_i(x_i^{(k)}) &\geq f_i(y^{(k)}) + a_i(y^{(k)})^\intercal(x_i^{(k)} - y^{(k)}) \geq f_i(y^{(k)}) - \left\|a_i(y^{(k)})\right\|\left\|x_i^{(k)} - y^{(k)}\right\| \\ &\geq f_i(y^{(k)}) - \varphi(\left\|x_i^{(k)} - \bar{x}^{(k)}\right\| + \left\|z^{(k)}\right\|) \geq f_i(y^{(k)}) - \varphi 3\beta, \end{aligned}$$

where we used $x_i^{(k)} - y^{(k)} = x_i^{(k)} - \bar{x}^{(k)} + z^{(k)}$ and $\|v_i^{(k)} - \bar{v}^{(k)}\| \leq \beta \Rightarrow \left\|v_i^{(k)} - v_j^{(k)}\right\| \leq 2\beta \Rightarrow \left\|x_i^{(k)} - x_j^{(k)}\right\| \leq 2\beta \Rightarrow \|x_i^{(k)} - \bar{x}^{(k)}\| \leq 2\beta$.

For any $y \in \mathcal{X}$, using Assumption 3.4.8 and the previous arguments, we get

$$\begin{aligned} f_i(y) &\geq f_i(x_i^{(k)}) + (a_i(x_i^{(k)}) + \nu)^\intercal(y - x_i^{(k)}) - \zeta \\ &\geq f_i(y^{(k)}) + (a_i(x_i^{(k)}) + \nu)^\intercal(y - x_i^{(i)}) - (3\varphi\beta + \zeta) \\ &= f_i(y^{(k)}) + (a_i(x_i^{(k)}) + \nu)^\intercal(y - y^{(k)} + y^{(k)} - x_i^{(k)}) - (3\varphi\beta + \zeta) \\ &\geq f_i(y^{(k)}) + (a_i(x_i^{(k)}) + \nu)^\intercal(y - y^{(k)}) - (\beta(6\varphi + 3\tau) + \zeta). \end{aligned}$$

Using the definition of an ε-subdifferential (2.4), this implies

$$(a_i(x_i^{(k)}) + \nu) \in \partial_{(\beta(6\varphi+3\tau)+\zeta)} f_i(y^{(k)}).$$

Summation of terms yields

$$f(y) \;\geq\; f(y^{(k)}) + \left(\sum_{i=1}^{n}(a_i(x_i^{(k)}) + \nu)\right)^{\mathsf{T}}(y - y^{(k)}) - n(\beta(6\varphi + \tau) + \zeta).$$

Based on Definition 2.1.5, this implies $\sum_{i=1}^{n}(a_i(x_i^{(k)})+\nu) \in \partial_{n(\beta(6\varphi+3\tau)+\zeta)} f(y^{(k)})$. □

We are now ready for the convergence theorem, which is based on the idea of interpreting (3.19) as an approximate subgradient algorithm.

Theorem 3.4.11. *Under Assumptions 3.2.1; 2.3.2; and 3.4.8, with the sequence* $\{x_1^{(k)}, ..., x_n^{(k)}\}_{k=0}^{\infty}$ *generated by* (3.19) *with*

$$\theta \geq \frac{\log(\beta) - \log(4n\xi(\beta + \alpha\varphi))}{\log(\gamma)};$$

$\left\|v_i^{(0)} - \bar{v}^{(0)}\right\| \leq \beta$*; and* $\beta/\alpha \leq \tau$*, we have:*
If $f^\star = -\infty$*, then*

$$\liminf_{k\to\infty} f(x_i^{(k)}) = -\infty,\ \forall i = 1, ..., n.$$

If $f^\star > -\infty$*, then*

$$\liminf_{k\to\infty} f(x_i^{(k)}) \;\leq\; f^\star + \alpha n(\varphi + \tau)^2/2 + n(\beta(9\varphi + 3\tau) + \zeta),\ \forall i \;=\; 1, ..., n.$$

Proof. From the definition of $y^{(k)}$ we have,

$$\begin{aligned} y^{(k+1)} =& \mathcal{P}_{\mathcal{X}}\left[\frac{1}{n}\sum_{i=1}^{n} v_i^{(k)}\right] = \mathcal{P}_{\mathcal{X}}\left[\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{n}[W^\theta]_{ij}u_j^{(k)}\right)\right] \\ =& \mathcal{P}_{\mathcal{X}}\left[\frac{1}{n}\sum_{i=1}^{n}\left(x_i^{(k)} - \alpha a_i(x_i^{(k)})\right)\right] = \mathcal{P}_{\mathcal{X}}\left[y^{(k)} + z^{(k)} - \frac{\alpha}{n}\sum_{i=1}^{n} a_i(x_i^{(k)})\right] \\ =& \mathcal{P}_{\mathcal{X}}\left[y^{(k)} - \frac{\alpha}{n}\sum_{i=1}^{n}\left(a_i(x_i^{(k)}) - \frac{z^{(k)}}{\alpha}\right)\right]. \end{aligned}$$

From Lemma 3.4.3, we know that $\left\|v_i^{(k)} - \bar{v}^{(k)}\right\| \leq \beta$ for all $i = 1, .., n$ and all $k \geq 0$. Furthermore, from Lemma 3.4.10, we know that $\sum_{i=1}^{n}(a_i(x_i^{(k)}) - z^{(k)}/\alpha) \in$

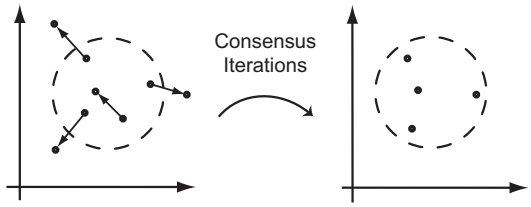


Figure 3.2: Each agent takes a step in the direction of its own subgradient. The initial iterates are within a β-ball indicated by the dashed circle. After the update, the iterates are outside the β-ball. Consensus iterations make sure that the iterates are within a β-ball.

$\partial_{n(\beta(6\varphi+3\tau)+\zeta)} f(y^{(k)})$ for all $k \geq 1$, since $\|z^{(k)}/\alpha\| \leq \beta/\alpha \leq \tau$ by assumption. In addition, we know that $\left\|\sum_{i=1}^{n}(a_i(x_i^{(k)}) - z^{(k)}/\alpha)\right\| \leq n(\varphi+\tau)$. Hence, $y^{(k)}$ is updated according to an approximate subgradient method and from Nedić (2002, Proposition 4.1) we have

$$\liminf_{k\to\infty} f(y^{(k)}) = -\infty, \text{ if } f^\star = -\infty$$

and

$$\liminf_{k\to\infty} f(y^{(k)}) \leq f^\star + \left(\alpha(n(\varphi+\tau))^2\right)/(2n) + n(\beta(6\varphi+3\tau)+\zeta), \text{ if } f^\star > -\infty.$$

Finally, we have

$$\begin{aligned}\liminf_{k\to\infty} f(x_i^{(k)}) &\leq \liminf_{k\to\infty} f(y^{(k)}) + n\beta 3\varphi \\ &\leq f^\star + \alpha n(\varphi+\tau)^2/2 + n(\beta(9\varphi+3\tau)+\zeta),\ \forall i = 1,...,n, k \geq 0.\end{aligned}$$

□

Remark 3.4.12. The assumption $\beta \leq \alpha\tau$ in Theorem 3.4.11 may seem restrictive, but it can be fulfilled with a fixed number of consensus negotiations according. The number of needed consensus iterations is given by Lemma 3.4.3.

Remark 3.4.13. The initial conditions in Theorem 1 and Theorem 2, $\left\|x_i^{(0)} - \bar{x}^{(0)}\right\| \leq \beta$ and $\left\|v_i^{(0)} - \bar{v}^{(0)}\right\| \leq \beta$, respectively, can be fulfilled with sufficiently many consensus negotiations before starting the algorithm. Another simple alternative is to set $x_i^{(0)} = x_j^{(0)}, \forall i,j$ and $v_i^{(0)} = v_j^{(0)}, \forall i,j$, respectively.

3.5 Numerical Results for MISM and CSM

In this section, we compare the MISM and the CSM. First, we define the objective function.

Objective Function

We use the following convex function as prototype

$$f_i(x) = (x - \check{x}_i)^\intercal Q_i (x - \check{x}_i)/2,$$

where Q_i is positive semi-definite. We quantize and integrate the gradient $\nabla f_i(x)$ to get a convex non-differentiable function. If Q_i is diagonal, as well as positive semi-definite, then the gradient is decoupled. We use the following quantization function, $q_i : \mathbb{R} \to \mathbb{R}$,

$$q_i([x]_j) = \beta_i(2l+1)/2, \quad \beta_i l \leq [x]_j \leq \beta_i(l+1),$$

where $\beta_i > 0$ is a fixed constant. We can now define our objective function

$$\tilde{f}_i(x) = \sum_{j=1}^{n} \int_0^{[x]_j} [Q_i]_{jj}(q_i(s) - [\check{x}_i]_j)ds. \tag{3.30}$$

We will now show that each term in the sum is convex, and for this purpose we let $g(x)$ denote such a term and $h(s)$ denote the integrand, where x and s are scalars. Pick $x, y, \alpha \in \mathbb{R}$ such that $x < y$ and $0 < \alpha < 1$, and let $z = \alpha x + (1-\alpha)y$. According to the definition, the function $g(x)$ is convex if $g(\alpha x + (1-\alpha)y) \leq \alpha g(x) + (1-\alpha)g(y)$. Note that

$$\int_x^z h(s)ds \leq (1-\alpha)(y-x) \max_{s\in[x,z]} h(s) \text{ and } \alpha(y-x) \min_{s\in[z,y]} h(s) \leq \int_z^y h(s)ds,$$

and since $h(s)$ is monotonically increasing we have

$$\frac{\int_x^z h(s)ds}{(1-\alpha)(y-x)} \leq \frac{\int_z^y h(s)ds}{\alpha(y-x)}.$$

Using this inequality, we arrive at

$$\begin{aligned}
\alpha g(x) + (1-\alpha)g(y) &= \alpha \int_0^x h(s)ds + (1-\alpha)\int_0^y h(s)ds \\
&= \alpha \int_0^z h(s)ds + (1-\alpha)\int_0^z h(s)ds - \alpha\int_x^z h(s)ds + (1-\alpha)\int_z^y h(s)ds \\
&= g(\alpha x + (1-\alpha)y) - \alpha\int_x^z h(s)ds + (1-\alpha)\int_z^y h(s)ds \\
&\geq g(\alpha x + (1-\alpha)y),
\end{aligned}$$

which establishes that the terms $g(x)$ are convex. Due to its construction, $\tilde{f}_i(x)$ is piecewise linear, and it is also convex since it is a sum of convex functions. An example of $\tilde{f}(x)$ is shown in Fig. B.1 on page 177.

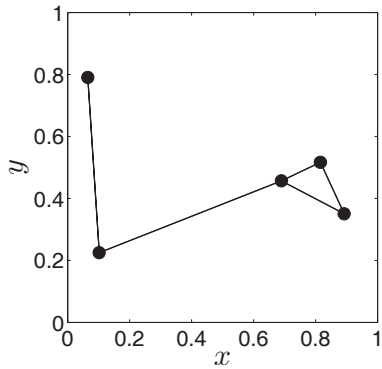


Figure 3.3: The topology corresponding to the optimization problem (3.31).

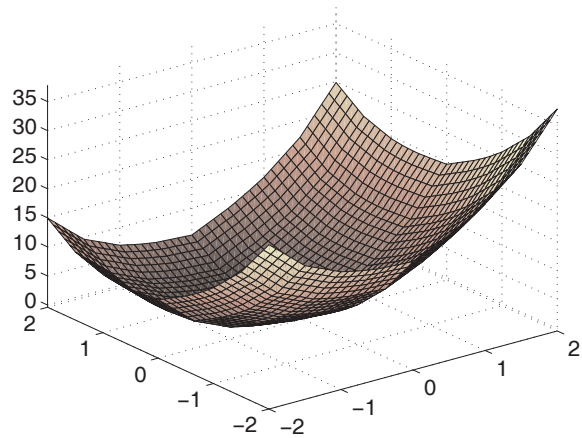


Figure 3.4: The objective function $f(x) = \sum_{i=1}^{5} \tilde{f}_i(x)$ where the parameters for $\tilde{f}_i$ are given in Table B.2 on page 178. Note that the function is convex and non-smooth.

Numerical Results

We consider the following toy problem: a 5-node network with the topology as illustrated by Fig. 3.3; each node or agent in the network corresponds to an objective function $\tilde{f}_i$ which are specified in (3.30) with parameters from Table B.2 on page 178. The sum of the objective functions $f(x) = \sum_{i=1}^{5} \tilde{f}(x)$ is illustrated in Fig. 3.4. We solve the following problem

$$\begin{array}{ll} \underset{x\in\mathbb{R}^2}{\text{minimize}} & \sum_{i=1}^{5} \tilde{f}_i(x) \\ \text{subject to} & -5 \leq [x]_1 \leq 5 \\ & -5 \leq [x]_2 \leq 5. \end{array} \tag{3.31}$$

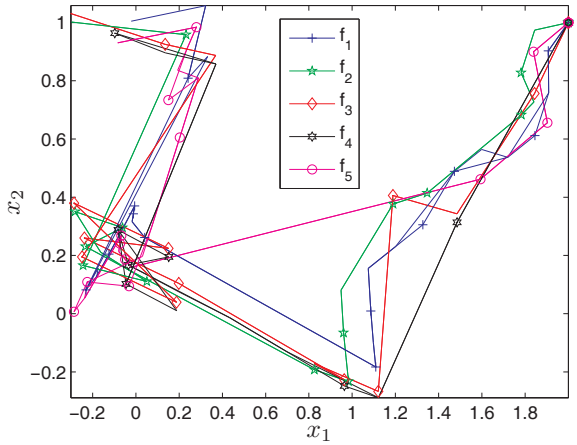


Figure 3.5: Trajectories of the iterates for each node for the MISM with $\alpha = 0.025$. The initial point is $(2, 1)$.

From Fig. 3.4 and the fact that $f(x)$ is convex, it is obvious that the optimal point will not change if we let $\mathcal{X} = \mathbb{R}^2$, and thus we can use an unconstrained solver for finding the optimal point. We find $x^\star = \begin{pmatrix} -1.4143 \cdot 10^{-12} & 0.5 \end{pmatrix}^\mathsf{T}$ using `fminsearch` in Matlab, which is a Nelder-Mead simplex algorithm (see, e.g., Lagarias et al. (1998)) based solver. It is not guaranteed to be the exact optimizer, but it is close enough for our purposes here.

First, we run the MISM algorithm with optimal probability transition matrix according to (2.11), where the numerical values are given in (B.1). We use the stepsize $\alpha = 0.025$, which was found to be a good choice by direct experimentation. The results for each node are shown in Fig. 3.5 and the normalized aggregate node error, $\sum_{i=1}^{5} \|f^\star - f(x_i^{(k)})\|/5$, are shown in Fig. 3.6. The stochastic nature of the MISM is very clear from both figures.

Second, we run the CSM algorithm with optimal consensus matrix according to (2.10), where the numerical values are given in (B.2). We use the stepsize $\alpha = 0.025$, which was found to be a good choice by direct experimentation. The results for each node are shown in Fig. 3.7 and Fig. 3.8. The normalized aggregate node error, $\sum_{i=1}^{5} \|f^\star - f(x_i^{(k)})\|/5$, versus subgradient iteration number, for different values of θ are shown in Fig. 3.9. The same error for different θ versus total number of iterations (subgradient iterations *plus* consensus iterations) are shown in Fig. 3.10. In this figure, we see that the consensus iterations slow down the algorithm extensively. This indicates that several consensus iterations should only be used if a more accurate solution is desired and the extra convergence time does not matter.

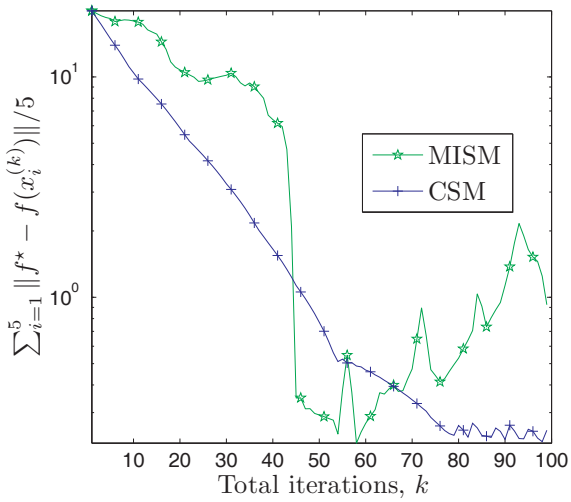


Figure 3.6: Comparison between the MISM with $\alpha = 0.025$ and the CSM with $\alpha = 0.025$ and $\theta = 1$.

3.6 Non-Smooth Center-Free Resource Allocation

We consider the following resource allocation problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} & \sum_{i=1}^{n} x_i = x_{\text{tot}}, \end{array} \tag{3.32}$$

with the functions $f_i : \mathbb{R} \to \mathbb{R}$ and $x = \begin{pmatrix} x_1 & \dots & x_n \end{pmatrix}^\mathsf{T}$. Let $f(x) = \sum_{i=1}^{n} f_i([x]_i)$, and denote the optimal value with $f^\star$ and the optimal set with $\mathcal{X}^\star = \{x \in \mathbb{R}^n \,|\, f(x) = f^\star\}$.

This is a special case of (3.1), with a special structure of the feasible set $\mathcal{X}$ (constant sum). The sum constraint could be decoupled using dual decomposition. The optimization problem would then decompose into n subproblems and a dual master program. The drawback with that approach, as Ho et al. (1980) point out, is that the iterates are only feasible at optimality. In some applications, were the iterative process has to be stopped before optimality is reached (if it is reached at all), and feasible iterates are necessary, dual relaxation will not work. Instead we could use the method proposed by Ho et al. (1980) and further developed by Xiao and Boyd (2006); see Section 2.6.3. However, this method requires the objective functions to be strictly convex as well as some other curvature constraints. We will develop an algorithm where the objective functions in (3.32) do not need to be differentiable; convexity and some other technical assumptions will be sufficient.

We also note that (3.32) belong to the class of monotropic optimization problems. This class consists of optimization problems where the objective function is a sum

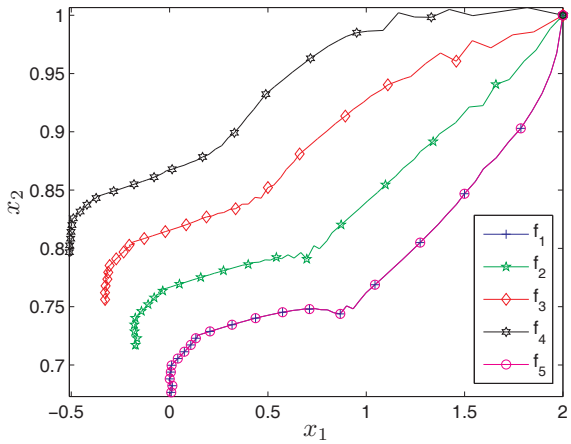


Figure 3.7: Trajectories of the iterates for each node for the CSM with $\alpha = 0.025$ and $\theta = 1$. Note that the trajectory of f_1 and f_5 overlap. This is not an error, instead, it is due to the values of W, see rows 1 and 5 in the matrix in (B.2), and the fact the initial point is the same for all nodes. The initial point is $(2, 1)$.

of linear functions composed with convex functions of a single variable (e.g., any quadratic function) subject to linear constraints; see, e.g., Rockafellar (1984).

Before presenting the algorithm, we need to introduce some notation and assumptions. We define the following sets, $\mathcal{B}_\eta = \{x \in \mathbb{R}^n \,|\, \|x\| \leq \eta\}$ with $\eta > 0$; $\mathcal{X} = \{x \in \mathbb{R}^n \mid \sum_{i=1}^n [x]_i = x_{\text{tot}}\}$; and $\bar{\mathcal{X}} = \{x \in \mathbb{R}^n \mid \sum_{i=1}^n [x]_i = 0\}$. Furthermore, we define the annulus $\mathcal{A}_\zeta^\eta = \{x \in \mathbb{R}^n | x \in \mathcal{X}^\star + \mathcal{B}_\eta \text{ and } x \notin \mathcal{X}^\star + \mathcal{B}_{\eta-\zeta}\}$, where $0 < \zeta < \eta$. The sets $\mathcal{X}^\star$, $\mathcal{X}^\star + \mathcal{B}_\eta$, and $\mathcal{A}_\eta^\zeta$ are illustrated in Fig. 3.11. Finally, we let $h(\eta, \zeta) = \inf_{x \in \mathcal{A}_\zeta^\eta} f(x)$.

Assumption 3.6.1. *i) The function $f : \mathbb{R}^n \to \mathbb{R}$ is convex and $\lim_{\|x\| \to \infty} f(x) = \infty$ (f is coercive). ii) The subgradients are bounded: $\sup\{\|a\| \,|\, a \in \partial f(x), x \in \mathcal{X}^\star + \mathcal{B}_\eta\} \leq \varphi$, with $\varphi > 0$. iii) There exist $\varepsilon > 0$ and $\eta > 0$ such that for all $x \in \mathcal{X}^\star + \mathcal{B}_\eta$, $a(x) \in \partial f(x)$, and $\|\nu\| \leq \tau$, the following holds $a(x) + \nu \in \partial_\varepsilon f(x)$. iv) There exists a $\zeta > 0$ such that $h(\eta, \zeta) - f^\star - \varepsilon > 0$.*

Remark 3.6.2. We have the following comments to Assumption 3.6.1. i) This assumption ensures that the set $\mathcal{X}^\star$ is nonempty and bounded (Rockafellar, 1970, Theorem 27.2). iii) This may seem to be a rather restrictive assumption, but it can be fulfilled for fixed η and arbitrarily small (fixed) ε by choosing τ appropriately; see Lemma 3.4.9. Furthermore, this is a curvature assumption, and such assumption is also needed for Ho et al. (1980) and Xiao and Boyd (2006). iv) This assumption guarantees that it is necessary to optimize: We can only hope to prove convergence to within ε close to the optimal value, and if this assumption is not fulfilled, we can achieve the best function value that we can hope for at the border of the feasible

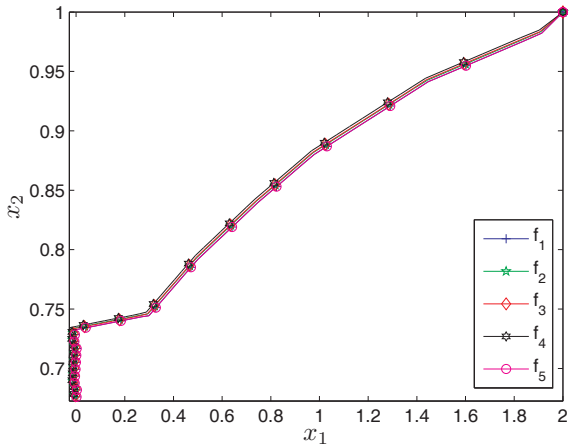


Figure 3.8: Trajectories of the iterates for each node for the CSM with $\alpha = 0.025$ and $\theta = 11$. The initial point is $(2, 1)$.

set, $\mathcal{X}^\star + \mathcal{B}_\eta$. In this case, the optimization algorithm will tell us that $x^\star \in \mathcal{X}^\star + \mathcal{B}_\eta$, which we already know by assumption.

The following lemma, where we establish that the projection on the set $\mathcal{X}$ can be written in different ways, will turn out to be very useful.

Lemma 3.6.3. *If $x \in \mathcal{X}$ and $y \in \mathbb{R}^n$ then $\mathcal{P}_{\mathcal{X}}[x+y] = x+\mathcal{P}_{\bar{\mathcal{X}}}[y] = x+\left(I - \frac{\mathbf{1}_n\mathbf{1}_n^\mathsf{T}}{n}\right) y$.*

Proof. The first equality holds, since for any $x \in \mathcal{X}$

$$\begin{array}{ll} \underset{z}{\text{minimize}} & \|x + y - z\| \\ \text{subject to} & \sum_{i=1}^n [z]_i = x_{\text{tot}} \end{array} \quad \Leftrightarrow \quad \begin{array}{ll} \underset{z'}{\text{minimize}} & \|y - z'\| \\ \text{subject to} & \sum_{i=1}^n [z']_i = 0, \end{array}$$

with $z^\star = z'^\star + x$. The second equality follows from the optimality conditions of constrained optimization, $b = \mathcal{P}_{\bar{\mathcal{X}}}[y]$ if and only if $(c - b)^\mathsf{T}(y - b) \leq 0$, $\forall c \in \bar{\mathcal{X}}$ (Bertsekas and Tsitsiklis, 1997, Proposition 3.2), since we have

$$\left(c - \left(I - \frac{\mathbf{1}_n\mathbf{1}_n^\mathsf{T}}{n}\right) y\right)^\mathsf{T} \left(y - \left(I - \frac{\mathbf{1}_n\mathbf{1}_n^\mathsf{T}}{n}\right) y\right) = 0,$$

if $c \in \bar{\mathcal{X}}$. □

If we let

$$U = I - \frac{\mathbf{1}_n\mathbf{1}_n^\mathsf{T}}{n},$$

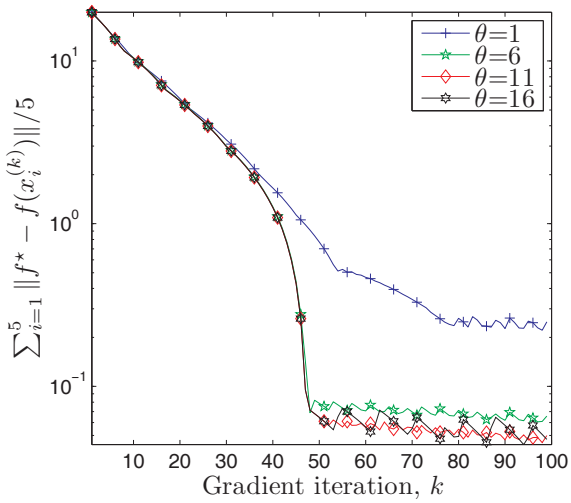


Figure 3.9: Average performance for different number of consensus negotiations. The x-axis denotes subgradient iterations.

then in view of Lemma 3.6.3, the projected subgradient method for (3.32) can be written as

$$x^{(k+1)} = \mathcal{P}_{\mathcal{X}}[x^{(k)} - \alpha a(x^{(k)})] = x^{(k)} - \alpha U a(x^{(k)}),$$

if $x^{(k)} \in \mathcal{X}$. But instead of using U directly, which requires all nodes to communicate directly with each other, let us use the following matrix. Define W according to the Metropolis-Hastings algorithm applied to the random walk on an undirected graph; see (2.9). The construction of W only requires local topology information. Let $\|W - \mathbf{1}_n\mathbf{1}_n^\intercal/n\| = \rho < 1$, and we have that $\lim_{\theta\to\infty} W^\theta = \mathbf{1}_n\mathbf{1}_n^\intercal/n$. Furthermore, let

$$V(\theta) = I - W^\theta.$$

Inspired by Ho et al. (1980) and similar to the algorithm in Section 3.4, we propose the following combination of subgradients and consensus iteration

$$x^{(k+1)} = x^{(k)} - \alpha V(\theta) a(x^{(k)}), \tag{3.33}$$

where $a(x^{(k)}) \in \partial f(x^{(k)})$ and $\alpha > 0$. Note that if we let $\theta \to \infty$, then the iteration becomes the standard projected subgradient method, since $\lim_{\theta\to\infty} V(\theta) = U$. Furthermore, also note that if $x^{(0)} \in \mathcal{X}$ then $x^{(k)} \in \mathcal{X}$ for all $k \geq 0$ due to the construction of $\alpha V(\theta)$. We denote this iteration the NCRM, Non-smooth Center-free Resource allocation Method.

The difference compared to the algorithm in Section 3.4 is that the each node has the same feasible $x^{(k)}$ at each time step. More specifically, this depends on

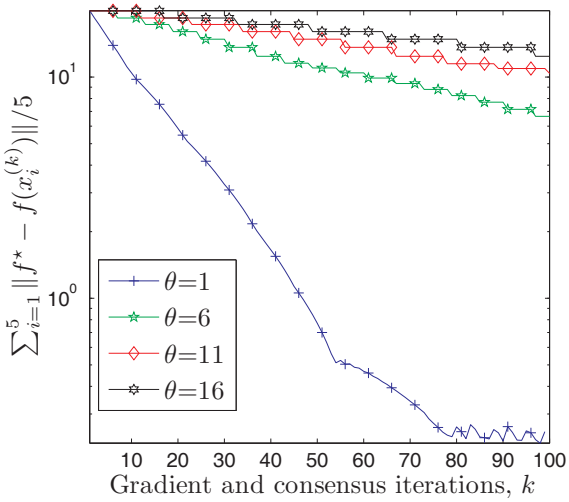
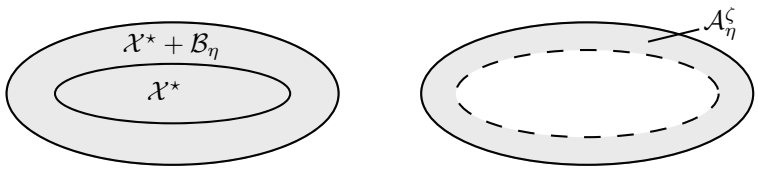


Figure 3.10: Performance plots for the CSM with $\alpha = 0.025$ and different θ. The x-axis denotes subgradient iterations *plus* consensus iterations.



(a) Illustration of the sets $\mathcal{X}^\star$ and $\mathcal{X}^\star + \mathcal{B}_\eta$.

(b) Illustration of the set $\mathcal{A}_\eta^\zeta$.

Figure 3.11: Illustration of the sets $\mathcal{X}^\star$, $\mathcal{X}^\star + \mathcal{B}_\eta$, and $\mathcal{A}_\eta^\zeta$.

that each node corresponds to one component in $x^{(k)}$ and that the sum of the components is constant. The difference between our algorithm and Ho et al. (1980) and Xiao and Boyd (2006) is that our algorithm does not rely on any differentiability assumptions.

3.6.1 Convergence Analysis

We have the following convergence result.

Theorem 3.6.4. *Under Assumption 3.6.1, the iteration* (3.33) *fulfills*

$$\lim_{k\to\infty} \operatorname*{dist}_{\mathcal{X}^\star + \mathcal{B}_{\omega(\varepsilon,\alpha)}} (x^{(k)}) = 0, \quad \omega(\varepsilon,\alpha) = \alpha\varphi + \max\left\{ \operatorname*{dist}_{\mathcal{X}^\star}(x) \,|\, x \in \mathcal{X}_{\frac{1}{2}\varphi^2\alpha+\varepsilon} \right\}, \quad (3.34)$$

where $\mathcal{X}_{\frac{1}{2}\varphi^2\alpha+\varepsilon} = \{x \in R^n | f(x) \leq f^\star + \frac{1}{2}\varphi^2\alpha + \varepsilon\}$ *if*

$$x^{(0)} \in \mathcal{X} \cap (\mathcal{X}^\star + \mathcal{B}_\eta),\ \theta > \frac{\log(\tau) - \log(\varphi)}{\log(\rho)},\ 0 < \alpha \leq \min\Big\{\frac{2(h(\eta,\zeta) - f^\star - \varepsilon)}{\varphi^2},\ \frac{\zeta}{\varphi}\Big\}$$

with $\zeta > 0$ *such that* $h(\eta,\zeta) - f^\star - \varepsilon > 0$. *Furthermore, we have that*

$$\lim_{(\varepsilon,\alpha)\downarrow 0} \omega(\varepsilon,\alpha) = 0. \tag{3.35}$$

Proof. The proof is divided into three parts. First, we show that the iteration can be interpreted as an ε-subgradient iteration if $x^{(k)} \in \mathcal{X} \cap (\mathcal{X}^\star + \mathcal{B}_\eta)$. Second, we show that the iterates always stay in this set. Third, we use standard ε-subgradient method convergence results. Let us start with the first part.

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - \alpha V(\theta)a(x^{(k)}) = x^{(k)} - \alpha U a(x^{(k)}) - \alpha(V(\theta) - U)a(x^{(k)}) && (3.36)\\ &= x^{(k)} - \alpha(V(\theta) - U)a(x^{(k)}) - \mathcal{P}_{\bar{\mathcal{X}}}[\alpha a(x^{(k)})] && (3.37)\\ &= \mathcal{P}_{\mathcal{X}}[x^{(k)} - \alpha(V(\theta) - U)a(x^{(k)}) - \alpha a(x^{(k)})] && (3.38)\\ &= \mathcal{P}_{\mathcal{X}}\left[x^{(k)} - \alpha\left(a(x^{(k)}) + \nu^{(k)}\right)\right], && (3.39) \end{aligned}$$

where (3.37) and (3.38) follows from Lemma 3.6.3 and that

$$x^{(k)} - \alpha(V(\theta) - U)a(x^{(k)}) \in \mathcal{X},$$

which follows from $x^{(k)} \in \mathcal{X}$ and $\mathbf{1}_n^\intercal(V(\theta) - U) = \mathbf{0}_n^\intercal$. By assumption

$$\|\nu^{(k)}\| = \|\alpha(V(\theta) - U)a(x^{(k)})\| \leq \|V - U\|^\theta \varphi \leq \rho^\theta \varphi \leq \tau,$$

which implies that $a(x^{(k)}) + \nu^{(k)} \in \partial_\varepsilon f(x^{(k)})$.

Now we proceed with the second part, where we show that the iterates always stay in the set $\mathcal{X} \cap (\mathcal{X}^\star + \mathcal{B}_\eta)$. If $x^{(k)} \in \mathcal{X}$, then

$$\mathbf{1}^T x^{(k+1)} = \mathbf{1}^T(x^{(k)} + \alpha V(\theta)) = \mathbf{1}^T x^{(k)},$$

and thus, $x^{(k+1)} \in \mathcal{X}$.

If $x^{(k)} \in (\mathcal{X}^\star + \mathcal{B}_{(\eta-\zeta)})$ then $\operatorname{dist}_{\mathcal{X}^\star}(x^{(k)}) \leq \eta - \zeta$ and $\|x^{(k+1)} - y\| \leq \|x^{(k+1)} - x^{(k)}\| + \|x^{(k)} - y\|$ for any $y \in \mathcal{X}^\star$. In particular, we have that

$$\operatorname*{dist}_{\mathcal{X}^\star}(x^{(k+1)}) \leq \|\alpha V(\theta)a(x^{(k)})\| + \eta - \zeta \leq \alpha\|V\|\varphi + \eta - \zeta \leq \alpha\varphi + \eta - \zeta \leq \eta,$$

since $\alpha\varphi - \zeta \leq 0$ by assumption. Thus, we have that $x^{(k+1)} \in (\mathcal{X}^\star + \mathcal{B}_\eta)$.

If $x^{(k)} \in \mathcal{A}_\eta^\zeta$, then, using the ε-subgradient inequality, for any $y \in \mathcal{X}^\star$,

$$\begin{aligned} \|x^{(k+1)} - y\|^2 &\leq \|x^{(k)} - y\|^2 - 2\alpha(f(x^{(k)}) - f(y) - \varepsilon) + \alpha^2\varphi^2\\ &\leq \|x^{(k)} - y\|^2 - 2\alpha(h(\eta,\zeta) - f^\star - \varepsilon) + \alpha^2\varphi^2 \leq \|x^{(k)} - y\|^2, \end{aligned}$$

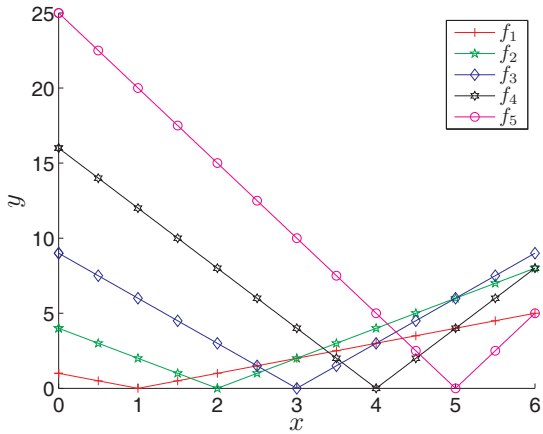


Figure 3.12: The terms in the objective function $f(x) = \sum_{i=1}^{5} i|x_i - i|$.

since $-2\alpha(h(\eta,\zeta) - f^\star - \varepsilon) + \alpha^2\varphi^2 \leq 0$ by assumption. We have that $x^{(k+1)} \in (\mathcal{X}^\star + \mathcal{B}_\eta)$. Hence, it follows that if $x^{(k)} \in \mathcal{X} \cap (\mathcal{X}^\star + \mathcal{B}_\eta)$ then $x^{(k+1)} \in \mathcal{X} \cap (\mathcal{X}^\star + \mathcal{B}_\eta)$.

Third, since $x^{(k)} \in \mathcal{X} \cap (\mathcal{X}^\star + \mathcal{B}_\eta)$ for all $k \geq 0$ and by Assumption 3.6.1 iii), we can interpret the iteration as a standard ε-subgradient algorithm. Then Kiwiel (2004, Theorem 4.1) gives us that

$$\lim_{k\to\infty} \operatorname*{dist}_{\mathcal{X}^\star + \mathcal{B}_{\omega(\varepsilon,\alpha)}} (x^{(k)}) = 0, \quad \omega(\varepsilon,\alpha) = \alpha\varphi + \max\left\{ \operatorname*{dist}_{\mathcal{X}^\star}(x) \,|\, x \in \mathcal{X}_{\frac{1}{2}\varphi^2\alpha+\varepsilon} \right\}, \quad (3.40)$$

where $\mathcal{X}_{\frac{1}{2}\varphi^2\alpha+\varepsilon} = \{x \in R^n | f(x) \leq f^\star + \frac{1}{2}\varphi^2\alpha + \varepsilon\}$, since $\|x^{(k+1)} - x^{(k)}\| \leq \alpha\varphi$ and $\|a(x^{(k)})\| \leq \varphi$. Finally, Kiwiel (2004, Lemma 2.4) yields

$$\lim_{(\varepsilon,\alpha)\downarrow 0} \omega(\varepsilon,\alpha) = 0.$$

□

Remark 3.6.5. The iterates will converge to a ball around the optimal set $\mathcal{X}^\star$. The radius of this ball can be made arbitrarily small by using sufficiently many consensus iterations θ and using a sufficiently small α.

3.7 Numerical Results for NCRM

We consider the same topology as in the previous section and we wish to solve the following problem.

$$\begin{aligned} \underset{x\in\mathbb{R}^5}{\text{minimize}} \quad & f(x) = \textstyle\sum_{i=1}^{5} i|[x]_i - i| \\ \text{subject to} \quad & \textstyle\sum_{i=1}^{5} [x]_i = 10, \end{aligned} \quad (3.41)$$

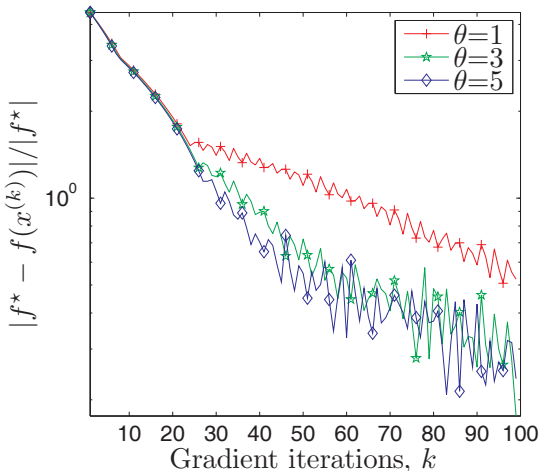


Figure 3.13: The performance of the NCRM. The aggregate error versus subgradient iterations.

where $f_i([x]_i) = i|x_i - i|$. It is clear that the optimizer of (3.41) is

$$x^\star = \begin{pmatrix} -4 & 2 & 3 & 4 & 5 \end{pmatrix}^\mathsf{T}.$$

We use the weighting matrix W from the previous section to construct $V(\theta) = I - W^\theta$. The performance of the optimization algorithm can is illustrated in Fig. 3.13. Furthermore, the deviation from the optimal point $x^\star$ for each node is shown in Fig. 3.14. Note that the sum is constant and that the iterates are feasible at all times. Finally, the performance of the NCRM compensated for the consensus iterations is shown in Fig. 3.15, i.e., the x-axis represents subgradient iterations *plus* consensus iterations. We see that many consensus iterations only seem to make sense if an accurate solution is necessary and the convergence time is not that important.

3.8 Summary

We have presented three optimization algorithms where there is no need for a central processing unit to whom each node should communicate its local subgradient.

First, we have proposed a novel randomized incremental subgradient method that is well suited for decentralized implementation in distributed systems. The algorithm is a generalization of the RISM and DISM due to Nedić and Bertsekas. These two algorithms can be recovered by choosing the transition probability matrix in a special way. The algorithm has been analyzed in detail with a convergence proof as well as a bound on the expected number of needed iterations to reach an *a priori* specified accuracy.

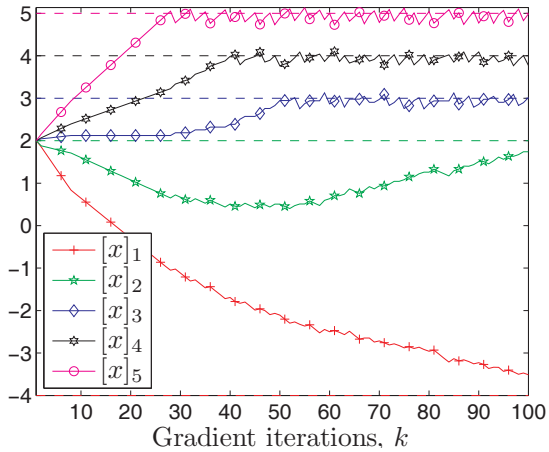


Figure 3.14: The performance of the NCRM. The individual error for each node from the optimal point, $x^\star = (-4\ 2\ 3\ 4\ 5)^\intercal$. The dashed lines represent the optimal allocation and the solid lines represent the iterates. Note that the iterates are feasible at all times.

Second, we have described an iterative subgradient-based method for solving coupled optimization problems in a distributed way given restrictions on the communication topology. In order to allow great flexibility in the information exchange architecture and distribute calculations, we combined the local subgradient updates with a consensus process. This means that computing agents can work in parallel with each other and use localized information exchange. For analysis purposes, we used results from consensus theory and employed approximate subgradient methods to study convergence properties of the proposed scheme. A connection is established between the number of consensus steps and the resulting level of optimality obtained by the subgradient updates.

Third, we have shown that under some mild technical assumptions, the center free algorithm introduced by Ho et al. (1980) can be extended to the subgradient case. Furthermore, the algorithm converges in an ε-sense if α is chosen sufficiently small and θ is chosen sufficiently large.

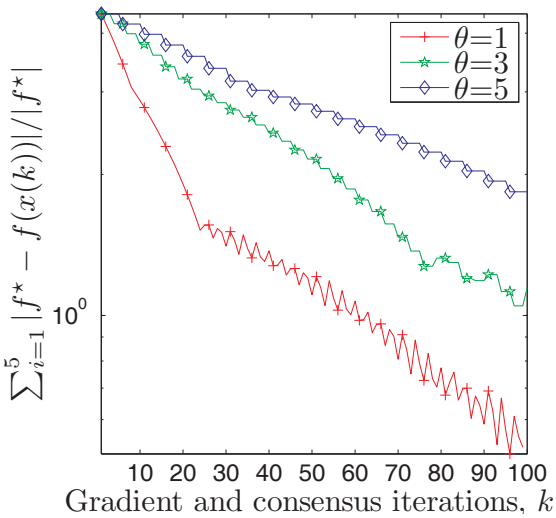


Figure 3.15: The performance of the NCRM. The aggregate error versus total iterations, i.e., subgradient iterations *plus* consensus iterations.

Chapter 4

Resource Allocation in Communication Networks

"*Resource allocation is tricky business.*"

P. J. Denning, The working set model for program behavior, ACM Symposium on Operating Systems Principles, 1967.

The performance of communication networks can be increased if the layers in the network are jointly optimized. In this chapter, we demonstrate and categorize some of the numerous ways in which decomposition and optimization techniques can be applied to engineer communication protocols, which jointly optimize the system-wide performance of the communication network. We derive specific results for two networking technologies. More specifically, we apply the decomposition techniques and optimization algorithms presented in Chapter 2 and Chapter 3, respectively, to devise cross-layer optimization algorithms for a frequency-division multiple access network and a spatial reuse time-division multiple access network. We call this *Network Utility Maximization* (NUM).

The chapter is organized as follows. We start with some general background in Section 4.1. In Section 4.2, we present two networking problems that will be solved. Section 4.3 discusses how the choice of the decomposition method can affect protocol properties and guide architectural considerations. Section 4.4 reviews optimization flow control for networks with fixed link capacities. In Section 4.5, we present a flowchart that is aimed to capture the essential steps in posing, decomposing, and solving network utility maximization problems. In Sections 4.6 and 4.7, we show how the decomposition principles presented in Section 2.7 can be used to devise distributed algorithms for two network scenarios. Considerable attention is given to the design of distributed mechanisms for allocating resources in the physical layer. Finally, Section 4.8 concludes the chapter with a summary.

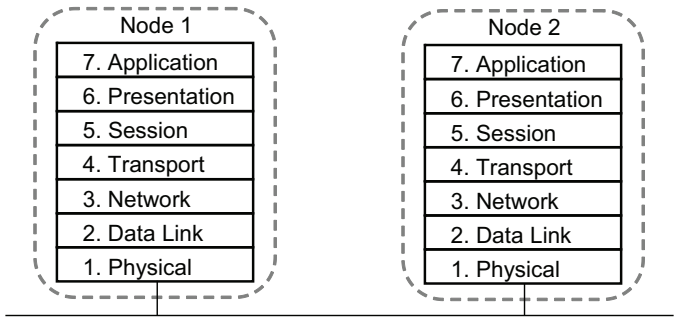


Figure 4.1: Node 1 and node 2 communicate with each other using communication protocols, which can be conceptualized with the OSI reference model. The OSI reference model contains seven layers, where higher level layers use the functionality of lower level layers.

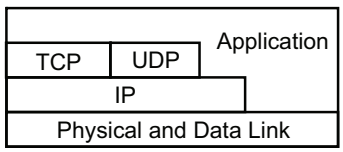


Figure 4.2: The current Internet architecture (Kawadia and Kumar, 2005).

4.1 Background

Network functionality is often divided into subsystems, so-called layers. These layers are logically stacked on top of each other, where the lower layers provide basic functionality for the higher layers. The higher layers can use the functionality provided by the lower layers without knowing anything about their inner workings. Moreover, the layered structure will make the design, and the maintenance of that design, much simpler. The most well-known protocol model is the OSI reference model (Zimmermann, 1980), which has seven layers; see Fig. 4.1 for an illustration. In addition, the structure of the current Internet architecture, which has been extremely successful, is illustrated in Fig. 4.2. Its predecessor, ARPANET, served as inspiration for the OSI model (the exact details of who inspired whom are, or at least were, controversial; see, e.g., Padlipsky (1982)).

In order to meet the increased demand for high-bandwidth network services, it has become increasingly important to develop network control mechanisms that utilize the full capabilities of all of the layers. However, in many network technologies, including optical and wireless networks, there is an inherent coupling between the layers. Adjusting the resource allocation in the physical layer changes the average link rates, influences the optimal routing, and alters the achievable network utility. Under such coupling, optimizing only within layers will not be enough to achieve

the optimal network performance, but the congestion control, routing, and physical layer control mechanisms need to be jointly designed. However, as mentioned in Chapter 1, there are also pitfalls with cross-layer optimization. Thus, such optimization has to be done with great care, to avoid unnecessary complexity. For example, Bush and Meyer (2002) argues that optimization is harmful since "optimization introduces complexity, and as well as introducing tighter coupling between components and layers."

4.1.1 Medium Access Protocols

In standard wired networks, the links correspond to physical cabling between the nodes and the link capacities are fixed. On the other hand, in a wireless network, the links are purely *logical* constructions representing the possibility of reliable communication between nodes. In many technologies, the link capacity depends on the amount of resources that is allocated to the link. In addition, the amount of resources allocated to all other links will also influence the link capacity, since the radio transmission medium is inherently a broadcast medium.

Due to the broadcast nature of the radio medium, we need a protocol that controls the access to the medium, such that the medium is used efficiently. There are many medium access protocols, which reside in the data link layer, and they can broadly be categorized into conflict-free and contention based (Rom and Sidi, 1990). In a conflict-free protocol, a transmission will not be interfered by another transmission, and this is typically achieved by dividing either time or frequency into non-overlapping parts. Furthermore, such channels are said to be *orthogonal*. It can also be useful to form hybrids, i.e., to combine time-division and frequency-division, which is done in the GSM standard (Ahlin and Zander, 1998).

When time is the entity being divided, the nodes are assigned time slots when they are allowed to transmit one at a time. Thus, when one node is transmitting, the other nodes wait for their turn. This is called Time-Division Multiple Access (TDMA), and the time slots form a schedule.

When the frequency spectrum is split into non-overlapping channels, each node is assigned a different channel. Hence, there will be no interference between simultaneous transmissions. This principle is denoted Frequency-Division Multiple Access (FDMA), and we will discuss it further in Section 4.6.

In these two principles, the capacity of the links can be changed. In TDMA, a link (or more precisely a node) can be assigned more slots or the slot length can be increased, which will give the link (node) more capacity. In FDMA, the bandwidth of the channel assigned to a link can be increased, which will increase the capacity of the link. It is also possible to improve these conflict-free protocols by using the fact that interference decrease with distance; nodes that are far apart can be assigned to transmit in the same time slot or frequency band without interfering with each other. In the TDMA case, this extension is called Spatial reuse Time-Division Multiple Access (STDMA), which we will discuss in Section 4.7.

In a contention based protocol, transmissions can interfere with each other,

and some conflict resolution algorithm is needed. A common technique is so-called Carrier Sense Multiple Access (CSMA), where a node that wish to transmit first listens to the channel to determine if it is idle; if the channel is idle, the node will start to transmit. We will use CSMA protocols in Chapter 6.

Finally, a more detailed introduction to TDMA, FDMA, and medium access protocols in general is given in, e.g., Rom and Sidi (1990).

4.1.2 An Optimization Approach

Recently, network utility maximization has emerged as a powerful framework for studying cross-layer issues; see, e.g., Kelly et al. (1998), Low and Lapsley (1999), Xiao et al. (2004), Chiang (2005), Lin and Shroff (2005), and Chiang et al. (2007). Although utility maximization is a mature subject in disciplines such as economics (e.g., Arrow and Hurwicz (1960)), its application to congestion control in communication networks is relatively recent; the papers by Kelly et al. (1998) and Low and Lapsley (1999) are considered to be pioneering. In addition, a related early work that uses optimization for flow control in communication networks is Golestaani (1979). The initial efforts in the networking literature focused on understanding various network control schemes (e.g., TCP/AQM variants) in the wired Internet as algorithms for solving a performance optimization problem, but it has also been used to engineer new congestion control schemes, notably TCP FAST (Wei et al., 2006).

The literature on utility maximization for networks with fixed link capacities is vast, and it is fair to say that there is now a relatively complete understanding of both equilibrium properties and of the dynamics of Internet congestion control (see, e.g., Srikant (2004) for a recent survey).

During the last couple of years, the basic model has been extended to include the effects of the wireless medium and a number of cross-layer optimal protocols have been suggested for different wireless technologies; see, e.g., Johansson and Johansson (2005), Wang et al. (2005), Chiang (2005), and Lin and Shroff (2005). However, one may argue that there has been limited innovation in terms of theoretical tools; almost all protocols have been designed using variations of the dual decomposition technique employed in the initial work by Low and Lapsley.

In this chapter, we demonstrate how the decomposition techniques presented in Chapter 2 suggest network architectures and protocols with different properties in terms of convergence speed, coordination overhead, and time-scales on which resource-updates should be carried out. Moreover, the techniques allow us to find distributed solutions to problems where the dual decomposition approach is not immediately applicable. The core of this chapter was published in Johansson et al. (2006*b*), but we note that similar ideas was put forward by D. P. Palomar and M. Chiang in the same issue (Palomar and Chiang, 2006).

The key contribution is to show how the alternative decomposition techniques can be applied to design novel distributed protocols for two wireless network technologies: FDMA networks with orthogonal channels and network-wide resource con-

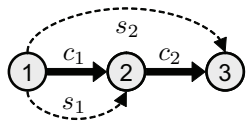


Figure 4.3: Example of a network with three nodes, two links, and two data-flows. The circles illustrate the nodes, the thick lines denote the directed links, and the dashed lines denote the data-flows.

straints, and wireless networks where the physical layer uses STDMA.

4.2 Network Utility Maximization

We consider a communication network formed by a set of nodes located at fixed positions. Each node is assumed to have infinite buffering capacity and can transmit, receive, and relay data to other nodes across communication links. The network performance then depends on the interplay between end-to-end rate selection in the transport layer, routing in the network layer, and resource allocation in the physical layer.

We model the network topology as a graph with χ directed links shared by η source-destination pairs; see Fig. 4.3 for an example network. To each source, we associate an increasing and strictly concave function $u_p(s_p)$, which measures the utility source p has of sending at rate s_p, and let $u(s) = \sum_{p=1}^{\eta} u_p(s_p)$ denote the aggregate utility. We assume that data is routed along fixed paths, represented by a routing matrix $R = [r_{lp}]$ with entries $r_{lp} = 1$ if source p uses link l and 0 otherwise. Moreover, we let c_l denote the capacity of link l. The optimal network operation can be found by solving the following NUM problem

$$\begin{aligned} \underset{s,c}{\text{maximize}} \quad & u(s) \\ \text{subject to} \quad & Rs \leq c, \quad s \in \mathcal{S}, \quad c \in \mathcal{C}, \end{aligned} \tag{4.1}$$

in the variables $s = \begin{pmatrix} s_1 & ... & s_\eta \end{pmatrix}^\intercal$ and $c = \begin{pmatrix} c_1 & ... & c_\chi \end{pmatrix}^\intercal$. In words, the problem is to maximize aggregate utility by jointly choosing s and c, subject to the constraint that the total traffic across links must be below the offered link capacities ($Rs \leq c$) and restrictions on the admissible end-to-end rates and link capacities ($s \in \mathcal{S}, c \in \mathcal{C}$). Specifically, the vector of end-to-end rates s must lie in a convex and closed set $\mathcal{S}$, typically on the form $\mathcal{S} = \{s|\ s_{\min} \leq s \leq s_{\max}\}$ or $\mathcal{S} = \{s|\ s_{\min} \leq s\}$, while the capacity vector must lie in the convex and closed multi-user capacity region $\mathcal{C}$ of the system. Any pair (s, c) that satisfies the constraints of (4.1) is said to be *feasible*, and corresponds to an admissible network operation. We make the following technical assumptions.

Assumption 4.2.1. *i) The network is connected. ii) The utility functions, $u_p(s_p)$,*

are strictly concave, differentiable, increasing, and $\lim_{s_p \to 0+} u_p(s_p) = -\infty$. *iii) The problem is feasible and a strictly interior point exists.*

The model (4.1) is rather abstract, as it hides the complexity of optimizing the capacity vector, e.g., allocating communications resources, such as time slots in a transmission schedule, transmission rates, powers, and bandwidths. Furthermore, the class of problems that fit into (4.1) is rather large, and to arrive at specific results, we will focus on two (still quite broad) particular cases of (4.1). These problem classes are practically relevant and have an underlying structure that allows us to go all the way to novel distributed solutions of the utility maximization problem.

Example 4.2.1 (Networks with Orthogonal Channels and Network-wide Resource Constraint)**.** We will now consider an FDMA network, where each link is assigned a unique frequency band of the available radio spectrum. Thus, we have a network-wide resource constraint (the available radio spectrum), which will complicate the optimization problem.

As previously mentioned, the model (4.1) is rather abstract and hides much of the complexity. In some cases, it is therefore more natural to be explicit about the resource dependence on the link rates, and use a model from the following class

$$\begin{array}{lll} \underset{s,\varphi}{\text{maximize}} & u(s) & \\ \text{subject to} & Rs \leq c(\varphi), & s_{\min} \leq s \\ & \sum_{l=1}^{\chi} \varphi_l \leq \varphi_{\text{tot}}, & 0 \leq \varphi, \end{array} \tag{4.2}$$

where s and φ have to be found to jointly maximize the aggregate utility, and where the link capacities are assumed to depend on a resource that has a network-wide constraint ($\sum_{l=1}^{\chi} \varphi_l \leq \varphi_{\text{tot}}$). This formulation can model a wireless or optical network that uses orthogonal channels and supports dynamic allocation of spectrum between transmitters. The total resource constraint complicates the problem since the resource allocation has to be coordinated across the whole network. The channel model could, for example, be the Shannon capacity (we will touch upon this in a bit more detail in Section 4.6.3)

$$c_l(\varphi_l) = \varphi_l \log\Big(1 + \frac{\beta_l}{\varphi_l}\Big),\ \varphi_l > 0.$$

In Section 4.6, we will demonstrate how the tools presented in Chapter 2 allow us to develop distributed algorithms for solving this NUM problem with network-wide resource constraints.

Example 4.2.2 (Cross-layer Optimized Scheduling in STDMA Wireless Networks)**.** In some cases, it is fruitful to keep the abstract formulation of (4.1) but to restrict the capacity region to have some specific structure. One interesting example is

$$\begin{array}{ll} \underset{s,c}{\text{maximize}} & u(s) \\ \text{subject to} & Rs \leq c, \qquad s_{\min} \leq s \\ & c \in \mathcal{C}_{\text{STDMA}}, \end{array} \tag{4.3}$$

where s and c have to be found to jointly maximize the aggregate utility, and where $\mathcal{C}_{\text{STDMA}}$ is a convex polytope, the convex hull of a finite set of points in $\mathbb{R}^{\chi}$. This seems to be very similar to the original problem, but the switch to a convex polytope as the feasible region, instead of only demanding the region to be convex, will prove to be crucial. This model captures networks that employ (possibly spatial-reuse) TDMA.

Section 4.7 demonstrates how the tools in Chapter 2 applied to this model suggest distributed mechanisms for constructing transmission schedules with multiple time slots.

4.3 Decomposition as Guiding Principle for Distributed Cross-layer Protocol Design

The approach advocated in this work relies on a mathematical network model that exposes the key interconnections between the network layers. Based on this model, we formulate the optimal network operation under user cooperation and cross-layer coordination as a global network utility maximization problem. To transform the centralized optimization problem into distributed protocols, we must find efficient ways for guiding different network elements towards the common goal. Inspiration for such coordination schemes can be found in mathematical decomposition techniques. Applying decomposition techniques to the global optimization problem allows us to identify critical information that needs to be communicated between nodes and across layers, and suggests how network elements should react to this information in order to attain the global optimum. In many cases, the underlying structure is such that these optimal rate and resource allocation schemes suggested by the decomposition schemes reside in separate networking layers. The layers are only loosely coupled via a set of critical control parameters. It turns out that the basic analysis suggests that these parameters are the Lagrange multipliers of the optimization problem.

The idea of decomposition applied to layers is not new, but has appeared in many papers during the last couple of years. However, the focus has been almost exclusively on using dual decomposition techniques. As we will see below, there are

several complementary techniques that give rise protocols with alternative properties, or allow us to find distributed algorithms to problems where the dual decomposition approach is not immediately applicable.

An underlying assumption of this work is that if a solution procedure of decomposition type has mathematical convergence, then it corresponds to a possible network architecture. Inspired by Holmberg (1995), we can take a step further and make a conjecture that a computationally efficient solution method corresponds to a better way of organizing the networking stack than what a less computationally efficient method does.

A crucial observation is that the link capacities c (or the resource allocation φ) are *complicating variables* in the sense that if the link capacities are fixed, (4.1) is simply an optimization flow control problem, which can be solved using the techniques in Kelly et al. (1998) and Low and Lapsley (1999). This is the topic of the next section.

4.4 Optimization Flow Control

Since optimization flow control will be a basic building block in our novel schemes, this section contains a brief review of the work in Kelly et al. (1998) and Low and Lapsley (1999). A key assumption is that the optimal bandwidth sharing mechanism solves the NUM problem

$$\begin{array}{ll} \underset{s}{\text{maximize}} & \sum_{p=1}^{\eta} u_p(s_p) \\ \text{subject to} & Rs \leq c, \qquad s \in \mathcal{S}, \end{array} \tag{4.4}$$

where the link capacity vector, c, is assumed to be fixed. A distributed solution to this problem can be derived via dual decomposition. Introducing Lagrange multipliers λ for the coupling constraints and forming the Lagrangian as in (2.28), one finds that the dual function

$$\mathrm{d}(\lambda) = \max_{s \in \mathcal{S}} \sum_{p=1}^{\eta} \left\{ u_p(s_p) - s_p \sum_{l=1}^{\chi} r_{lp} \lambda_l \right\} + \sum_{l=1}^{\chi} \lambda_l c_l$$

is separable in end-to-end rates s_p, and can be evaluated by letting sources optimize their rates individually based on the total congestion price along the end-to-end path, i.e., by letting

$$s_p = \arg\max_{z_p \in \mathcal{S}} u_p(z_p) - z_p \sum_{l=1}^{\chi} r_{lp} \lambda_l, \quad p = 1, ..., \eta. \tag{4.5}$$

Moreover, the dual problem can be solved by the projected gradient iteration

$$\lambda_l^{(k+1)} = \mathcal{P}_{\Lambda} \left[\lambda_l^{(k)} + \alpha^{(k)} \left(\sum_{p=1}^{\eta} r_{lp} s_p^{(k)} - c_l \right) \right], \quad l = 1, ..., \chi,$$

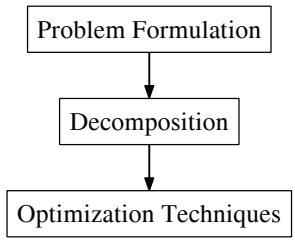


Figure 4.4: Overview of the flowchart, showing the three major blocks. The blocks are further explained in Fig. 4.5, Fig. 4.6, and Fig. 4.8.

where $\alpha^{(k)}$ is the stepsize. Note that links can update their congestion prices based on local information: if the traffic demand across link l exceeds capacity, the congestion price increases; otherwise it decreases. Convergence of the dual algorithm has been established in Low and Lapsley (1999). As argued in Low and Lapsley (1999) and Low (2003), the equilibrium points of a wide range of TCP protocols can be interpreted in terms of sources maximizing their marginal utilities (utilities minus resource costs). Furthermore, the link algorithms generate prices to align the sources' selfish strategies with the global optimum. Finally, different TCP/AQM variants are identified with different utility functions and laws for updating the link prices.

In general, the dynamics of the optimization flow control algorithm can either be placed in the source rate update, in the link price update, or in both. These variants are called primal, dual, and primal-dual algorithms, respectively. Note that this naming convention does not imply anything about the decomposition (if any) of the optimization problem.

4.5 How Should NUM Problems be Posed, Decomposed, and Solved?

The answer to that question will of course very much depend on the specifics of the problem. However, looking at the literature, it is possible to see some patterns. We have made an effort to categorize some of the existing approaches to solving NUM problems using the flowchart in Fig. 4.4. The flowchart consists of the following three blocks:

Problem Formulation The problem formulation is of paramount importance. The model has to capture all relevant phenomena while still being sufficiently simple; this is the classic fidelity versus tractability tradeoff. In addition, the difference between a good problem formulation and a bad one can be the difference that makes it solvable or not. Specifically, convex and decomposable optimization problems are desired. There is no simple recipe on how to devise

a good model, but we will outline some useful techniques. The details are visualized in Fig. 4.5.

Decomposition In order to engineer a decentralized algorithm, the optimization problem has to be split into several subproblems that can be solved in a distributed way using some coordination scheme. Most often, either a dual or primal decomposition technique is chosen; see Section 2.7. The details are shown in Fig. 4.6.

Optimization Techniques The resulting subproblems have to be solved in some way; numerous methods can of course be used, and we include the most common methods. The details are illustrated in Fig. 4.8.

To capture all decomposition approaches previously pursued in the literature, the flowchart would have to be huge, with the effect that it would be practically useless. Therefore, we are forced to compromise, and the flowchart we present captures, in our opinion, the most crucial steps and algorithmic approaches in solving NUM problems. The aim is to find patterns in the existing literature, to make it easier to get an overview what has been done, and to visualize the inherent steps in the NUM framework. The initial result can be seen at `http://www.networkutilitymaximization.org/`. Since it is web based, it is quite possible for the flowchart to evolve. However, it should not expand too much, since then the overview will be lost. We will now go into more details for each of the blocks in Fig. 4.4 and give examples on how the blocks should be interpreted, although we refer to the website for full details and numerous examples.

4.5.1 Problem Formulation

As previously mentioned, convex problems can be effectively solved, and therefore, it is rather natural to look for problem transformations and re-formulations that result in a problem which is convex in the optimization variables. We will now describe the most important steps in the problem formulation block; the steps are visualized in Fig. 4.5.

Change of Variables

One important trick used in the literature is to explore the log-change of variables from geometric programming. In particular, let us consider the following constraint, which is a high SINR approximation of the Shannon capacity (see, e.g., Proakis (2001)),

$$t_l \leq \varphi_l \log\left(\frac{G_{ll}P_l}{\sigma_l + \sum_{j\neq l} G_{lj}P_j}\right),$$

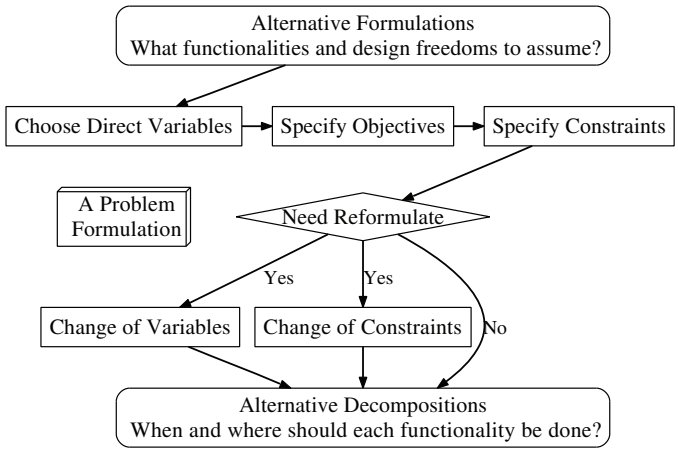


Figure 4.5: The steps in the problem formulation block from Fig. 4.4.

which is not convex in the variables (t_l, P). However, using the change-of-variables $P_l = e^{\tilde{P}_l}$, the constraint becomes (see, e.g., Chiang (2005))

$$t_l \leq -\varphi_l \log\left(\sigma_l G_{ll}^{-1} e^{-\tilde{P}_l} + \sum_{j \neq l} G_{ll}^{-1} G_{lj} e^{\tilde{P}_j - \tilde{P}_l}\right)$$

which is convex, since the log-sum-exp function is convex (Boyd and Vandenberghe, 2004).

Combined Variable and Constraint Transformations

In other situations, one might need to perform both constraint transformations and variable transformations to arrive at convex formulations. As an example, consider the constraint (the Shannon capacity)

$$t_l \leq \varphi_l \log\left(1 + \frac{G_{ll} P_l}{\sigma_l + \sum_{j \neq l} G_{lj} P_j}\right).$$

By taking the logarithms of both sides

$$\log(t_l) \leq \log\left(\varphi_l \log\left(1 + \frac{G_{ll} P_l}{\sigma_l + \sum_{j \neq l} G_{lj} P_j}\right)\right),$$

the variable transformation $t_l = e^{\tilde{t}_l}, P_l = e^{\tilde{P}_l}$ results in the constraint

$$\tilde{t}_l \leq \log\left(\varphi_l \log\left(1 + \frac{G_{ll} e^{\tilde{P}_l}}{\sigma_l + \sum_{j \neq l} G_{lj} e^{\tilde{P}_j}}\right)\right),$$

which is convex in the new variables $\tilde{t}_l$ and $\tilde{P}$ (Papandriopoulos et al., 2006).

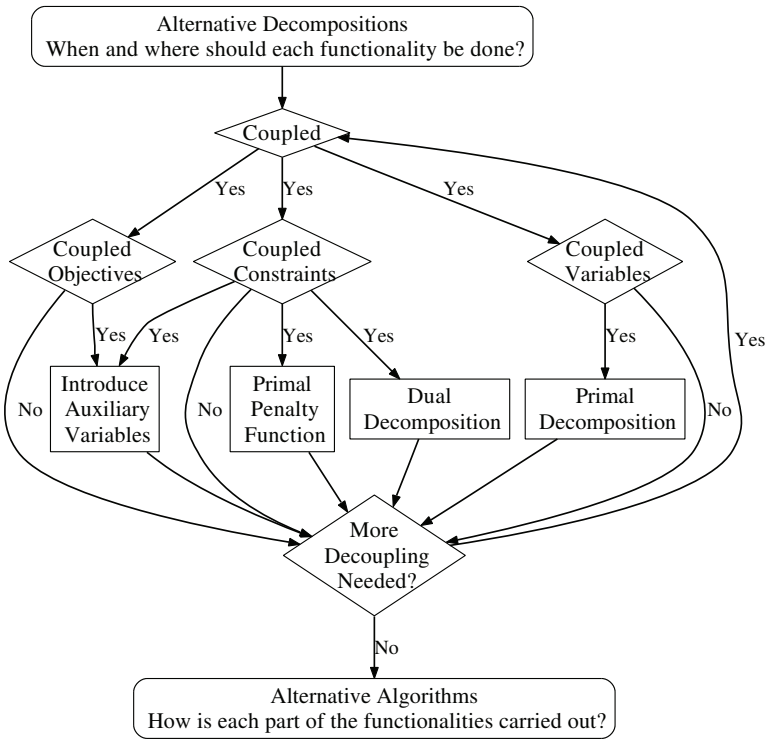


Figure 4.6: The steps in the decomposition block from Fig. 4.4.

4.5.2 Decomposition

Decomposition techniques allow us to decompose an optimization problem into subproblems. As mentioned in Section 2.7, there are basically two types of decomposition: primal and dual. Usually, primal and dual decomposition give rise to a master problem and one or several subproblems. The subproblems can then also be decomposed, resulting in a lower level master problem with its own subproblems; see Fig. 4.7 for an illustration. Thus, we can say that each decomposition adds another "level" into the optimization problem. Furthermore, each "level" will typically require at least one tuning parameter, and a multitude of parameters can make the resulting algorithm very time consuming to tune. Thus, it is wise to avoid unnecessary decompositions. We will now describe the most important steps in the decomposition block in Fig. 4.4.

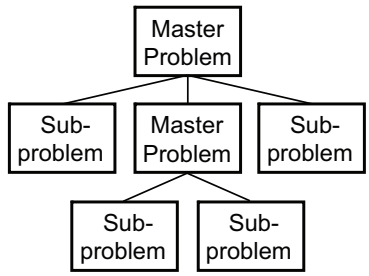


Figure 4.7: The optimization problem is decomposed into three levels: a master problem and three subproblems, where the middle subproblem is decomposed into one master problem and two subproblems.

Introduce Auxiliary Variables

To deal with optimization problems with complicated objective functions, such as

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \sum_i f_i(\sum_j x_{ij}) \\ \text{subject to} & x \in X, \end{array}$$

it is sometimes useful to introduce auxiliary variables, $z_i = \sum_j x_{ij}$, and consider the equivalent formulation

$$\begin{array}{ll} \underset{z_i,x}{\text{minimize}} & \sum_i f_i(z_i) \\ \text{subject to} & z_i = \sum_j x_{ij}, \quad x \in X. \end{array}$$

Combining such re-formulations with the mathematical decomposition techniques described in Section 2.7 can pave way for efficient distributed solutions.

Primal Penalty Functions

The idea with penalty functions is that a constraint is replaced by a penalty function (or barrier function). The penalty function becomes infinite when the constraint is not fulfilled, and the constraint will therefore be enforced. This is a method that can be used instead of or in combination with primal and dual decomposition. When penalty functions are used, a sequence of optimization problems typically has to be solved, where the penalty function is refined at each step, either using a predefined scheme or an adaptive scheme. More theoretical details can be found in, e.g., Bertsekas (1999, Section 4.1). For example, penalty functions can be used for congestion control (Srikant, 2004, Section 3.1): Instead of solving the optimization problem

$$\begin{array}{ll} \underset{s}{\text{maximize}} & \sum_{p=1}^{\eta} u_p(s_p) \\ \text{subject to} & Rs \leq c, \qquad s \in \mathcal{S}, \end{array}$$

we can solve the following approximate version

$$\begin{array}{ll} \underset{s}{\text{maximize}} & \sum_{p=1}^{\eta} u_p(s_p) - \sum_{l=1}^{\chi} \int_0^{[Rs]_l} f_l(y)dy \\ \text{subject to} & s \in \mathcal{S}, \end{array}$$

where f_l are penalty functions which are continuous and non-decreasing such that $\lim_{k\to\infty} \int_0^k f_l(y)dy = \infty$. The penalty functions f_l should be chosen such that we get close to the original problem, and an adaptive choice of f_l that achieves this is given in Srikant (2004).

Coupled Constraints

If we have the constraint $\sum_i A_i x_i \leq c$, then we can directly apply dual decomposition. Otherwise, we can introduce auxiliary variables, y_i, and we get

$$A_i x_i \leq y_i, \quad \sum_i y_i \leq c,$$

which we can decompose using primal decomposition.

Coupled Variables

A more general version of the coupled constraints problem is the following,

$$\sum_i A_i x_i \leq y, \quad y \in \mathcal{Y}. \tag{4.6}$$

In this case, we can either use primal decomposition directly on (4.6), or we can introduce auxiliary variables, y_i,

$$A_i x_i \leq y_i, \quad \sum_i y_i = y, \quad y \in \mathcal{Y},$$

which we the decompose using dual decomposition.

Primal vs. Dual decomposition

If we have constraints that can be relaxed (e.g., average power constraint or average capacity constraint), then dual decomposition can be used since the constraint is only fulfilled for optimal λ.

On the other hand, if the constraints need to be enforced strictly over all time (e.g., peak power constraint), then primal decomposition can be used, since the iterates always are feasible.

4.5.3 Optimization Techniques

The decomposition step leads to several subproblems, which need to be solved. Therefore, we have to go through the optimization block for all subproblems. We will now describe the most important steps in the optimization block in Fig. 4.4. The details are illustrated in Fig. 4.8.

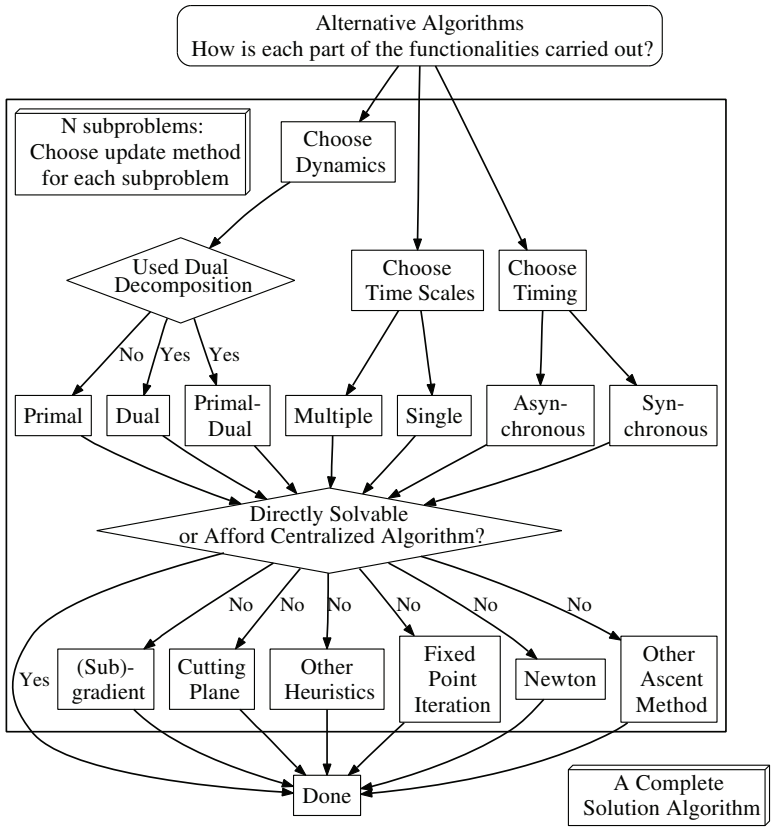


Figure 4.8: The details of the steps in the optimization block from Fig. 4.4. This block has to be applied to all subproblems.

Primal, Dual, and Primal-Dual

If the NUM problem contains a flow control part, it is customary to classify the flow control algorithm as either, primal, dual, or primal-dual, depending on where the dynamics are located; see Section 4.4. Note that this naming convention has nothing to do with primal and dual in its mathematical programming sense, which makes it a bit confusing.

Time scales

Decomposition typically leads to nested optimization problems, with at least one inner problem and an outer master problem. Separation of time scales (the inner problem is solved to optimality) usually enables us to show guaranteed convergence properties for solution algorithms. Using a more refined analysis, it may be possible

to show convergence even without separation of time scales.

Synchronous vs. Asynchronous

The subproblem can either be solved in a synchronous or an asynchronous fashion. The synchronous case is much simpler to analyze, whereas the asynchronous case is more general. Maintaining synchronous operations in a large network is demanding, and most real protocols will run asynchronously. Therefore, it is comforting if the analysis also guarantees proper operation of the algorithm under asynchronous operation as well. More details on asynchronous algorithms can be found in Bertsekas and Tsitsiklis (1997).

(Sub)gradient Methods

Subgradient methods, see Chapter 3, can be used to minimize (maximize) non-differentiable convex (concave) functions. Subgradient methods do not require much assumptions on the objective function, and they are simple to implement in a decentralized fashion. To guarantee convergence to an optimal solution, the stepsize typically needs to be diminishing. However, a diminishing stepsize can be hard to implement, since it requires synchronization, and the progress will be very slow after a while. Therefore, a fixed stepsize is often used. Furthermore, in some cases, more exotic variations, e.g., the incremental subgradient method and its variants (see Chapter 3), can be used. These variants can be useful when the application has requirements on the communication pattern of the optimization algorithm, e.g., peer-to-peer communication.

If a gradient is available, a constant stepsize may also guarantee convergence to the optimal objective value under some Lipschitz continuity assumptions. Subgradient and gradient methods are very common in the NUM literature.

Cutting Plane

Cutting plane methods can solve constrained non-differentiable convex optimization problems. These methods rely on polyhedral approximations of the optimization problem, which means that the original problem is relaxed to a linear program with a huge number of constraints (in the general case). Furthermore, the method starts with a subset of these constraints and add more during its execution (Elhedhli et al., 2001). Cutting plane methods are typically centralized, but can converge faster than subgradient methods.

Other Heuristics

For integer programming problems, such as scheduling, some heuristic algorithms may be used. In many cases, the optimal scheduling solution can only be achieved in a centralized approach. However, in some cases, distributed and suboptimal algorithms with minimum performance guarantees can be found (e.g., distributed

weighted maximum matching (Lotker et al., 2007)). Such algorithms are often denoted *approximation algorithms* .

Fixed Point Iteration

Some algorithms can be interpreted as a general iteration converging to a fixed point with desirable properties. For example, although the distributed power control problem of Foschini and Miljanic (1993) can be posed and analyzed as a linear program, it is more naturally viewed as a fixed point iteration.

Newton

Newton's method is a weighted ascent algorithm: the gradient is multiplied with the inverse of the Hessian ($H^{-1}\nabla f(x)$ where $H = \nabla^2 f(x)$). The method enjoys rapid convergence, especially for quadratic like objective functions. However, the objective function needs to be twice differentiable and the Hessian needs to be inverted (if it is invertible), which typically is costly to compute and hard to make in a distributed fashion. See, e.g., Bertsekas (1999, Section 1.2) for more details.

Other Ascent Method

It is possible to use a method that increases the objective value at each iteration without belonging to one of the mentioned classes.

4.6 Networks with Orthogonal Channels and Network-wide Resource Constraint

As a first application, we consider the design of utility maximizing protocols for systems with orthogonal channels and a global resource constraint in the physical layer. Hence, we would like to solve the problem (4.2), which we now present once more for easy reference

$$\begin{aligned} \underset{s,\varphi}{\text{maximize}} \quad & u(s) \\ \text{subject to} \quad & Rs \leq c(\varphi), \qquad s_{\min} \leq s \\ & \textstyle\sum_{l=1}^{\chi} \varphi_l \leq \varphi_{\text{tot}}, \quad 0 \leq \varphi. \end{aligned} \tag{4.7}$$

4.6.1 Optimality Conditions and Decomposition Approaches

The problem formulation (4.7) can be rewritten using the properties of the optimal point under the following assumptions

Assumption 4.6.1. *i) The channel capacities $c_l(\varphi_l)$ are strictly concave, twice differentiable, increasing, and $c_l(0) = 0$. ii) Each row and column of the routing*

matrix R contains at least one positive entry. iii) $s_{\min} > 0$. iv) The problem is strictly feasible, i.e., $\sum_{l=1}^{\chi}\left(c_l^{-1}(\sum_{p=1}^{\eta} R_{lp}(s_{\min}+\varepsilon))\right) < \varphi_{tot}$ for some $\varepsilon > 0$.

The routing matrix assumption implies that all sources are assumed to be sending and all links are used by at least one source. The assumption on $s_{\min}$ and φ_{tot} means that φ_{tot} is large enough to allow all sources to send just above the minimum sending rate using the links indicated by the routing matrix. The optimal point can then be characterized as follows.

Lemma 4.6.2. *The optimal point, $(s^\star, \varphi^\star, \lambda^\star)$, to (4.7) under Assumptions 4.2.1 and 4.6.1 is characterized by*

$$\begin{cases} \sum_l \varphi_l^\star = \varphi_{tot} & \\ Rs^\star = c(\varphi^\star) & \\ \lambda_l^\star c_l'(\varphi_l^\star) = \varsigma^\star, & l = 1, ..., \chi \\ \varphi_l^\star \geq \varphi_{\min} > 0, & l = 1, ..., \chi \\ s_p^\star \geq s_{\min}, & p = 1, ..., \eta. \end{cases} \tag{4.8}$$

Proof. We will use the KKT conditions (see Section 2.7.1) to characterize the optimal point. The Lagrangian is

$$\begin{aligned} \mathrm{L} = &\sum_{p=1}^{\eta} u_p(s_p) + \sum_{l=1}^{\chi} \lambda_l \Big(c_l(\varphi_l) - \sum_{p=1}^{\eta} R_{lp} s_p \Big) + \\ &\varsigma \Big(\varphi_{\text{tot}} - \sum_{l=1}^{\chi} \varphi_l \Big) + \sum_{p=1}^{\eta} (s_p - s_{\min}) m_p + \sum_{l=1}^{\chi} \varphi_l n_l. \end{aligned}$$

The KKT conditions for the optimal point are

$$\begin{cases} \frac{\partial \mathrm{L}}{\partial s_p} = u_p'(s_p^\star) - \sum_{l=1}^{\chi} \lambda_l^\star R_{lp} + m_p^\star = 0 & \\ \frac{\partial \mathrm{L}}{\partial \varphi_l} = \lambda_l^\star c_l'(\varphi_l^\star) - \varsigma^\star + n_l^\star = 0 & \\ \sum_l \varphi_l^\star - \varphi_{\text{tot}} \leq 0, & (\sum_{l=1}^{\chi} \varphi_l^\star - \varphi_{\text{tot}})\, \varsigma^\star = 0 \\ \sum_{p=1}^{\eta} R_{lp} s_p^\star - c_l(\varphi_l^\star) \leq 0, & \varsigma^\star \geq 0 \\ \left(\sum_{p=1}^{\eta} R_{lp} s_p^\star - c_l(\varphi_l^\star) \right) \lambda_l^\star = 0, & \lambda_l^\star \geq 0 \\ s_p^\star \geq s_{\min}, & (s_{\min} - s_p^\star) m_p^\star = 0 \\ m_p^\star \geq 0, & \varphi_l^\star \geq 0 \\ \varphi_l^\star n_l = 0, & n_l^\star \geq 0. \end{cases}$$

Elimination of $m_s^\star$ and $n_l^\star$ yields

$$\begin{cases} \sum_l \varphi_l^\star - \varphi_{\text{tot}} \leq 0, & (\sum_l \varphi_l^\star - \varphi_{\text{tot}})\,\varsigma^\star = 0 \\ \sum_{p=1}^{\eta} R_{lp} s_p^\star - c_l(\varphi_l^\star) \leq 0, & \varsigma^\star \geq 0 \\ \left(\sum_{p=1}^{\eta} R_{lp} s_p^\star - c_l(\varphi_l^\star)\right) \lambda_l^\star = 0, & \lambda_l^\star \geq 0 \\ \sum_{l=1}^{\chi} \lambda_l^\star R_{lp} - u_p'(s_p^\star) \geq 0, & s_p^\star \geq s_{\min} \\ \varsigma^\star - \lambda_l^\star c_l'(\varphi_l^\star) \geq 0, & \varphi_l^\star(\varsigma^\star - \lambda_l^\star c_l'(\varphi_l^\star)) = 0 \\ (s_{\min} - s_p^\star)\left(\sum_{l=1}^{\chi} \lambda_l^\star R_{lp} - u_p'(s_p^\star)\right) = 0, & \varphi_l^\star \geq 0. \end{cases}$$

Note that each row in R has positive entries by assumption. Combining this with $\sum_{p=1}^{\eta} R_{lp} s_p^\star - c_l(\varphi_l^\star) \leq 0$ and $s_p^\star \geq s_{\min}$ yields $\varphi_l^\star > 0$ for all $l = 1, ..., \chi$. Similarly, combining the assumption that the columns in R have positive entries with $\sum_{l=1}^{\chi} \lambda_l^\star R_{lp} - u_p'(s_p^\star) \geq 0$ and $u_p'(s_p^\star) > 0$ (u_p is strictly concave and increasing by assumption), gives that at least one $\lambda_l^\star$ is positive. Hence, $\varsigma^\star - \lambda_l^\star c_l'(\varphi_l^\star) \geq 0$ and $c_l'(\varphi_l^\star) > 0$ (c_l is strictly concave and increasing by assumption) give that $\varsigma^\star > 0$. Now, $\varphi_l^\star > 0$ and $\varphi_l^\star(\varsigma^\star - \lambda_l^\star c_l'(\varphi_l^\star)) = 0$ imply that $\lambda_l^\star > 0$ and $\varsigma^\star = \lambda_l^\star c_l'(\varphi_l^\star)$ for all $l = 1, ..., \chi$. Furthermore, $\varsigma^\star > 0$ and $\lambda_l^\star > 0$ for all $l = 1, ..., \chi$ give that $\sum_{l=1}^{\chi} \varphi_l^\star = \varphi_{\text{tot}}$ and $Rs^\star = c(\varphi^\star)$. Finally, since $\sum_{p=1}^{\eta} R_{lp} s_p \geq \sum_{p=1}^{\eta} R_{lp} s_{\min}$, we have that $\varphi_l \geq \varphi_{\min} = c_l^{-1}(\sum_{p=1}^{\eta} R_{lp} s_{\min})$ for all $l = 1, ..., \chi$. □

Remark 4.6.3. Thus, in the optimal solution to (4.7), the common resource is fully utilized, all links are bottlenecks and the marginal link revenues $\lambda_l^\star c_l'(\varphi_l^\star)$ are equal.

One consequence of this is that it is possible to use an equivalent problem

$$\begin{aligned} \underset{x,\varphi}{\text{maximize}} \quad & u(s) \\ \text{subject to} \quad & Rs \leq c(\varphi), \qquad s_{\min} \leq s \leq s_{\max} \\ & \textstyle\sum_l \varphi_l = \varphi_{\text{tot}}, \quad \varphi_{\min} \leq \varphi, \end{aligned} \tag{4.9}$$

which has some special properties that we can exploit. More specifically, when we decompose (4.9), we will get resource allocation subproblems, which we can solve with the center-free algorithms from Section 2.6.3. The problems (4.7) and (4.9) are equivalent in the sense that they share the same optimal solution. The crucial change is that $\sum_l \varphi_l \leq \varphi_{\text{tot}}$ has been changed to $\sum_l \varphi_l = \varphi_{\text{tot}}$. Moreover, some bounds have been introduced; the upper bound, $s_{\max}$, on s and the lower bound $\varphi_{\min}$, on φ are technical conditions that do not change the optimal point, but make the analysis simpler. This simpler problem will be solved using two approaches, dual and primal decomposition. We also introduce the convex and compact sets $\mathcal{S} = \{s | s_{\min} \leq s \leq s_{\max}\}$ and $\Phi = \{\varphi | \varphi_{\min} \leq \varphi, \sum_l \varphi_l = \varphi_{\text{tot}}\}$.

Dual Approach

Introducing Lagrange multipliers, λ_l, $l = 1, \ldots, \chi$, for the capacity constraints in (4.9), we form the partial Lagrangian

$$\mathrm{L}(s, \varphi, \lambda) = \sum_{p=1}^{\eta} u_p(s_p) - \lambda^\intercal (Rs - c(\varphi))$$

and the associated dual function

$$\mathrm{d}(\lambda) = \underbrace{\max_{s \in \mathcal{S}} \left\{ \sum_{p=1}^{\eta} u_p(s_p) - \lambda^\intercal Rs \right\}}_{\text{Network subproblem}} + \underbrace{\max_{\varphi \in \Phi} \lambda^\intercal c(\varphi).}_{\substack{\text{Resoure allocation} \\ \text{subproblem}}} \tag{4.10}$$

Thus, the dual function decomposes into a network subproblem and a resource allocation subproblem. The network subproblem is identical to the end-to-end rate allocation problem in optimization flow control (4.5), while distributed solutions for the resource allocation second subproblem will be developed in Section 4.6.2. Since the link capacities are assumed to be strictly concave, the partial Lagrangian is strictly concave in (s, φ) and the dual function is differentiable (Bertsekas, 1999, Proposition 6.1.1) and

$$\nabla \mathrm{d}(\lambda) = c(\varphi^\star(\lambda)) - Rs^\star(\lambda).$$

Similarly to optimization flow control, the dual problem (2.30) can be solved using the projected (sub)gradient iteration

$$\lambda_l^{(k+1)} = \mathcal{P}_\Lambda \left[\lambda_l^{(k)} - \alpha^{(k)} (c_l(\varphi_l^{(k)}) - [Rs^{(k)}]_l) \right], \quad l = 1, ..., \chi, \tag{4.11}$$

where the stepsizes, $\alpha^{(k)}$, fulfill (2.19) and $\mathcal{P}_\Lambda[\cdot]$ denotes projection on the positive orthant. This update can be carried out locally by links based on their current excess capacities. The optimization problem (4.9) can be solved with Algorithm 6, as shown in the following lemma.

Lemma 4.6.4. *Under Assumptions 4.2.1 and 4.6.1, Algorithm 6 with stepsizes fulfilling* (2.19)*, e.g., $\alpha^{(k)} = 1/k$, converges to the optimal solution to* (4.7)*, i.e.,*

$$\lim_{k \to \infty} u(s^{(k)}) = u^\star.$$

Proof. The subgradient[1] is

$$\nabla g(\lambda) = c(\varphi^\star(\lambda)) - Rs^\star(\lambda),$$

[1] The subgradient is unique in this case, and therefore, it is also the gradient.

Algorithm 6 Dual

1: Let $k \leftarrow 0$, $\lambda^{(0)} \geq 0$.
2: **loop**
3: Solve the network subproblem using (4.5) for $\lambda^{(k)}$ to get $s^{(k)}$.
4: Solve the resource allocation subproblem using an algorithm from Section 4.6.2 for $\lambda^{(k)}$ to get $\varphi^{(k)}$.
5: Get $\lambda^{(k+1)}$ via (4.11).
6: Let $k \leftarrow k + 1$.
7: **end loop**

where s is in the interval $s_{\min} \leq s \leq c(\varphi_{tot})$ and φ is in the interval $\varphi_{\min} \leq \varphi \leq \varphi_{tot}$. This implies that the subgradient is bounded as follows

$$||\nabla g(\lambda)||_2 \leq ||c(\varphi_{\text{tot}}) + Rc(\varphi_{\text{tot}})||_2.$$

Furthermore, we also know that the optimal set is nonempty, since the feasible set is compact. With the assumptions 4.2.1 and 4.6.1, convergence now follows from Bertsekas et al. (2003, Proposition 8.2.6). □

Note that the optimal resource allocation and source rates can be found in parallel, but the optimal solutions to both subproblems should be found before the dual variables are updated. From a practical perspective, even disregarding potential synchronization issues, this approach has the disadvantage that resource allocations have to be done at a fast time-scale and that the resource allocation algorithm (at least in the most basic analysis) has to be executed to optimality before the dual variables can be updated.

Primal Approach

As an alternative, we apply primal decomposition and re-write (4.9) as

$$\begin{aligned} \underset{\varphi}{\text{maximize}} \quad & \nu(\varphi) \\ \text{subject to} \quad & \varphi \in \Phi, \end{aligned} \tag{4.12}$$

where we have introduced

$$\nu(\varphi) = \max_{s \in \mathcal{S}} \{u(s) | Rs \leq c(\varphi)\}. \tag{4.13}$$

Note that $\nu(\varphi)$ is simply the optimal network utility that can be achieved by optimization flow control under resource allocation φ. Consequently, to evaluate $\nu(\varphi)$ we can fix the resource allocation and execute the distributed optimization flow control from Section 4.4 until convergence.

Before attempting to solve the problem (4.12), we will establish some basic properties of $\nu(\varphi)$.

Lemma 4.6.5. *Under assumptions 4.2.1 and 4.6.1, $\nu(\varphi)$ is concave and a subgradient, $a(\varphi)$, of $\nu(\varphi)$ at φ is given by*

$$a(\varphi) = \begin{pmatrix} \lambda_1 c_1'(\varphi_1) & \cdots & \lambda_\chi c_\chi'(\varphi_\chi) \end{pmatrix},$$

where λ_l are optimal Lagrange multipliers for the capacity constraints in (4.13).

Proof. By strong duality,

$$\nu(\varphi) = \min_{\lambda \geq 0} \max_{s \in \mathcal{S}} \sum_{p=1}^{\eta} (u_p(s_p) - s_p q_p) + \sum_{l=1}^{\chi} \lambda_l c_l(\varphi_l) = \min_{\lambda \geq 0} \tilde{g}(s(\lambda)) + \sum_{l=1}^{\chi} \lambda_l c_l(\varphi_l),$$

with $q_p = \sum_{l=1}^{\chi} r_{lp} \lambda_l$ and some function $\tilde{g}$. Thus, since $\nu(\varphi)$ is the pointwise infimum of concave functions, it is concave. Let $\lambda^\star$ be the optimal Lagrange multipliers for a resource allocation vector φ. For any other resource allocation $\tilde{\varphi}$, it holds that

$$\nu(\tilde{\varphi}) \leq \max_{s \in \mathcal{S}} \left\{ \sum_{p=1}^{\eta} u_p(s_p) - s_p q_p^\star + \sum_{l=1}^{\chi} \lambda_l^\star c_l(\tilde{\varphi}_l) \right\} \leq \nu(\varphi) + \sum_{l=1}^{\chi} \lambda_l^\star c_l'(\varphi_l)(\tilde{\varphi}_l - \varphi_l),$$

with $q_p^\star = \sum_{l=1}^{\chi} r_{lp} \lambda_l^\star$. This, by the definition of a subgradient, concludes the proof. □

Since a subgradient of ν is available, it is natural to use a projected subgradient algorithm

$$\varphi^{(k+1)} = \mathcal{P}_\Phi \left[\varphi^{(k)} + \alpha^{(k)} a(\varphi^{(k)}) \right], \tag{4.14}$$

with diminishing stepsize, $\alpha^{(k)}$, fulfilling (2.19). Here $\mathcal{P}_\Phi[\cdot]$ denotes *distributed projection* on the set Φ, i.e., solves the following projection problem

$$\begin{array}{ll} \underset{\varphi}{\text{maximize}} & -||\varphi - \varphi^0||_2^2 = -\sum_{l=1}^{\chi} (\varphi_l - \varphi_l^0)^2 \\ \text{subject to} & \varphi \in \Phi, \end{array} \tag{4.15}$$

in a distributed fashion. This is a non-trivial problem. However, the projection problem has a separable concave objective function since $-||\varphi - \varphi^0||_2^2 = \sum_{l=1}^{\chi} -(\varphi_l - \varphi_l^0)^2$, and the projection problem can be solved using the techniques in Section 4.6.2.

Theorem 4.6.6. *Under assumptions 4.2.1 and 4.6.1, Algorithm 7 with stepsizes according to* (2.19)*, e.g., $\alpha^{(k)} = 1/k$, converges to the optimal solution to* (4.7)*, i.e.,*

$$\lim_{k \to \infty} u(s^{(k)}) = u^\star \text{ and } \lim_{k \to \infty} \varphi^{(k)} = \varphi^\star.$$

Algorithm 7 Primal

1: Let $\varphi^{(0)} > 0$ and $k \leftarrow 0$.
2: **loop**
3: Fix the resource allocation and let the congestion control scheme solve (4.13) for $\varphi^{(k)}$ to get a subgradient.
4: Get $\varphi^{(k+1)}$ via (4.14) and perform the distributed projection using an algorithm from Section 4.6.2.
5: Let $k \leftarrow k + 1$.
6: **end loop**

Proof. The subgradient is given by $a(\varphi) = \begin{pmatrix} \lambda_1 c_1'(\varphi_1) & \cdots & \lambda_\chi c_\chi'(\varphi_\chi) \end{pmatrix}$. Since $a(\varphi)$ is continuous in φ, bounded for every φ, and φ lies in a compact set, the subgradient is bounded by (the finite value) $\max_{\varphi_{\min} \leq \varphi \leq \varphi_{\text{tot}}} ||a(\varphi)||_2$. We also have that the optimal set is non-empty, since the feasible set is compact. Together with the assumptions 4.2.1 and 4.6.1, convergence now follows from Bertsekas et al. (2003, Proposition 8.2.6). □

The primal method relies on solving the optimization flow problem on a fast time-scale and performing incremental resource updates in an ascent direction of the total network utility on a slower time-scale. The source rate and link price updates are carried out in a distributed way, similarly to optimization flow control. As we will show next, the resource update can be performed in a distributed manner that only relies on communication and resource exchanges between direct neighbors. Put together, this results in a completely distributed algorithm for the network utility maximization problem.

4.6.2 Solving the Resource Allocation Subproblem

Two problems remain to be solved: the resource allocation subproblem in the dual approach (4.10),

$$\begin{aligned} \underset{\varphi}{\text{maximize}} \quad & \textstyle\sum_{l=1}^{\chi} \lambda_l c_l(\varphi_l) \\ \text{subject to} \quad & \varphi \in \Phi, \end{aligned}$$

and the distributed projection in the primal approach (4.15),

$$\begin{aligned} \underset{\varphi}{\text{maximize}} \quad & \textstyle\sum_{l=1}^{\chi} -(\varphi_l - \varphi_l^0)^2 \\ \text{subject to} \quad & \varphi \in \Phi. \end{aligned}$$

Both problems are on the form of (2.12), which once again is

$$\begin{array}{ll} \underset{\varphi}{\text{maximize}} & \sum_{l=1}^{\chi} f_l(\varphi_l) \\ \text{subject to} & \sum_{l=1}^{\chi} \varphi_l = \varphi_{\text{tot}} \\ & \varphi_l \geq \varphi_{\min},\ l = 1, ..., \chi. \end{array}$$

The properties of the optimal solution to this problem was presented in Section 2.5.1. We present two algorithms that solve (2.12): the weighted gradient approach and the direct negotiation approach.

Weighted Gradient Approach

The algorithms in Ho et al. (1980) and Xiao and Boyd (2006) solve (2.12) under the assumptions that $f_l(\cdot)$ are concave, twice continuously differentiable, with the second derivative bounded below and above, $m_l \leq f_l''(\varphi_l) < 0$, with m_l known, as we have seen in Section 2.6.3. The resource updates rely on nearest neighbor communication only, and can be written in vector form as

$$\varphi^{(k+1)} = \varphi^{(k)} + W \nabla f(\varphi^{(k)}). \tag{4.16}$$

The limitation that links should only be allowed to communicate and exchange resources with its neighbors turns up as a sparsity constraint on W; see Section 2.6.3. A simple way of guaranteeing convergence is to pick W according to the Metropolis-Hastings scheme; see Section 2.6.3. Note that W can be constructed with only limited topology information.

The idea that nodes should give away resources to neighbors that have better use for them is very intuitive, and the idea has been used before. In for example Antal et al. (1998), a similar scheme, based on heuristics, is suggested to be used in dynamic synchronous transfer mode optical networks. In this scheme, the nodes have tokens that give them right to a certain bandwidth. The nodes are suggested to transfer tokens to a neighbor that have more use of the bandwidth; more precisely, a token is transferred if the expression $|$(priority of node i) $\cdot$ (free channels of node $i+1$) $-$ (priority of node $i+1$) $\cdot$ (free channels of node i)$|$ decreases by a token transfer.

Direct Negotiation Approach

As an alternative, the resource allocation problem (2.12) can be solved via direct negotiation. This scheme requires the network to be ordered in a ring structure (or, in fact, any other structure providing order). A similar structure is also needed for determining a starting point that guarantees non-negativity of the iterates of the weighted gradient method (Ho et al., 1980).

The approach is based on the so-called water-filling method; see, e.g., Boyd and Vandenberghe (2004, Section 5.5.3). The idea is to use the optimality conditions

given in Section 2.5.1. More specifically, we use an algorithm that finds $\psi^\star$ (see Section 2.5.1). We define the function $h_l : \mathbb{R} \to \mathbb{R}$ as

$$h_l(\psi) = \begin{cases} \varphi_{\text{tot}}, & \text{if } \psi < f_l'(\varphi_{\text{tot}}) \\ (f_l')^{-1}(\psi), & \text{if } f_l'(\varphi_{\text{tot}}) \leq \psi < f_l(0) \\ 0, & \text{if } f_l'(0) \leq \psi. \end{cases}$$

Note that h_l is a continuous function, decreasing in ψ, and that the inverse of $f_l'(\varphi_l)$ is well-defined since $f_l(\varphi_l)$ is strictly concave. Also, introduce $h(\psi) = \sum_{l=1}^{\chi} h_l(\psi)$, which is a sum of decreasing continuous functions, and hence, h is also continuous and decreasing. Now, the water-filling method finds $\psi^\star$ such that

$$h(\psi^\star) = \varphi_{\text{tot}}. \tag{4.17}$$

A lower bound for $\psi^\star$ is $\underline{\psi} = \min_l f_l'(\varphi_{\text{tot}})$, and an upper bound is $\overline{\psi} = \max_l f_l'(0)$, so $\underline{\psi} \leq \psi^\star \leq \overline{\psi}$. Since $h(\psi)$ is continuous and decreasing in ψ, we can use binary search to find $\psi^\star$ that fulfills (4.17). Start with setting $\psi \leftarrow (\overline{\psi} + \underline{\psi})/2$. If $h(\psi)$ is below φ_{tot}, then set the upper bound to ψ, i.e., $\overline{\psi} \leftarrow \psi$. But if $h(\psi)$ is above φ_{tot} then set the lower bound to ψ, i.e., $\underline{\psi} \leftarrow \psi$. Repeat until (4.17) is satisfied with desired accuracy.

The algorithm can be executed in a distributed way: one node takes the lead and announces the initial value of ψ. The node forwards the current value of ψ and the sum of claimed resources to the next node in the structure. After a complete cycle, the total sum of requested resources is available and the search interval for $\psi^\star$ can be cut in half as above. The process is repeated until the desired accuracy is achieved.

4.6.3 Numerical Results

To illustrate the performance of the approaches, we apply the dual and the primal algorithms to a sample 8-node network, which is connected and was used in Johansson and Johansson (2005). The network topology is shown in Fig. 4.10. Furthermore, the channel capacities are chosen to be the Shannon capacities (see, e.g., Proakis (2001)) and the resource is channel bandwidth:

$$c_l(\varphi_l) = \begin{cases} \varphi_l \log\left(1 + \frac{\beta_l}{\varphi_l}\right), & \varphi_l > 0 \\ 0, & \varphi_l = 0. \end{cases}$$

We have that $\lim_{\varphi_l \to 0} c_l(\varphi_l) = 0$ and $c_l''(\varphi_l) = -\beta_l^2/(\varphi_l(\varphi_l + \beta)^2) < 0$ when $\varphi_l > 0$. In addition, $c_l'(\varphi_l) = \log(1 + \beta_l/\varphi_l) - \beta_l/(\varphi_l + \beta_l)$, which is always positive when $\varphi_l > 0$. To see this, introduce $y = 1 + \beta_l/\varphi_l$, and we have that

$$\log(1 + \beta_l/\varphi_l) - \beta_l/(\varphi_l + \beta_l) > 0,\ \varphi_l > 0 \Leftrightarrow \log(y)y - y + 1 > 0,\ y > 1,$$

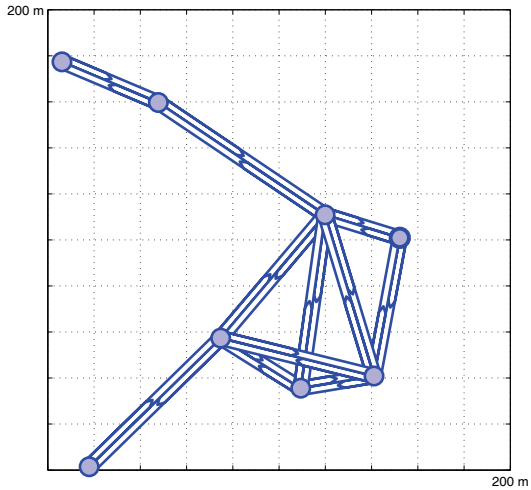


Figure 4.9: The node placement of the example network. The filled circles denote the nodes and the lines indicate the links in the network.

where the last expression is clearly positive for large y. When $y = 1$, we have that $\log(y)y - y + 1 = 0$, and from $\frac{\partial}{\partial y}\log(y)y - y + 1 = \log(y) > 0$, $y > 1$, we see that $\log(y)y - y + 1 > 0$ when $y > 1$. Thus, c_l fulfill Assumption 4.6.1.

Furthermore, the utility functions are logarithmic,

$$u_p(s_p) = \log(s_p),$$

so they fulfill Assumption 4.2.1.

Finally, the routing matrix is assumed to be not of full rank, but each row and column contain at least one positive entry; the rest of Assumption 4.6.1 is fulfilled by the construction of the example. Thus, both Assumptions 4.2.1 and 4.6.1 are fulfilled for this example.

Solving the optimization problem with the primal and the dual algorithms yields the results in Figure 4.10. Both algorithms are converging to the optimal point. The dual algorithm descends in a smoother fashion, which can be explained by that the dual function is differentiable. However, the dual algorithm seems to be sensitive to the starting point and the initial stepsize (this cannot be seen in the current plot): if either of these is badly adjusted the convergence rate can be significantly reduced. Moreover, the iterates are only primal feasible in the limit, which is illustrated in Fig. 4.11. We can see in the figure that the source rates are never feasible, and at least one link is operating 15 % over available capacity. The primal algorithm exhibits the typical subgradient method behavior, i.e., the descent is not monotone. However, this algorithm seems to be more robust with respect to the starting point and initial stepsize, and the iterates are always primal feasible.

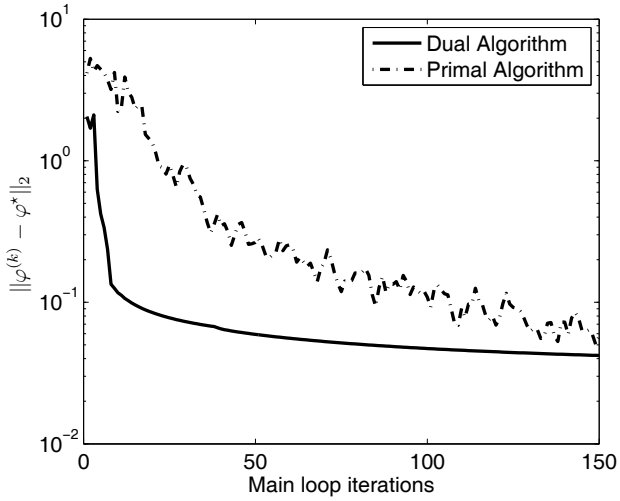


Figure 4.10: The norm of the resource allocation minus the optimal resource allocation versus main loop iterations for the dual and primal algorithms.

4.7 Cross-layer Optimized Scheduling in STDMA Wireless Networks

Our second application considers network utility maximization of wireless networks that employ STDMA. As previously mentioned, STDMA is a collision-free access scheme that allows spatially separated radio terminals to transmit simultaneously when the interference they incur on each other is not too severe; see, e.g., Grönkvist (2005). We will consider a particular instance of STDMA networks that offers a single communication rate, c_{tgt}, to all links that obey both primary and secondary interference constraints. The *primary interference* constraints require that a node communicates with at most one other node at a time (these constraints are typically imposed by the underlying technology, e.g., nodes equipped with omnidirectional antennas and no multi-user detectors). The *secondary interference* constraints require that the signal-to-interference and noise ratios at the receivers of active links exceed a target value

$$\frac{G_{ll}P_l}{\left(\sigma_l + \sum_{j\neq l} G_{lj}P_j\right)} \geq \gamma_{\text{tgt}}.$$

Here, P_l is the transmit power of link l, σ_l is the thermal noise power at the receiver of link l, and G_{lj} denotes the effective power gain from the transmitter of link j to the receiver of link l. We say that a set of links is a *feasible transmission group* if there is a power allocation such that all links in the set obey the primary and secondary interference constraints. Associated to each feasible transmission group

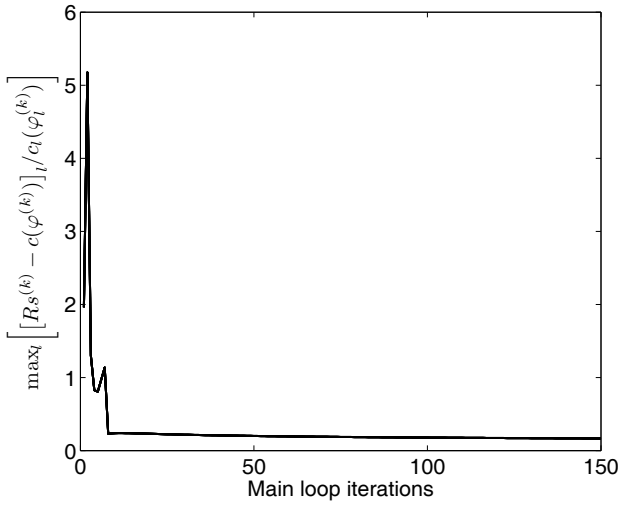


Figure 4.11: The maximum normalized difference between used capacities and available capacities versus main loop iterations for the dual algorithm. The curve will be at zero when the source rates are feasible. At least one link is consistently operating 15 % over available capacity.

is a feasible link capacity vector where $c_l = c_{\text{tgt}}$ for all links l in the group and $c_l = 0$ otherwise. By time-sharing over a large number of time slots, we can achieve *any* average link capacity vector in the convex hull of the feasible link capacity vectors.

4.7.1 Decomposition Approaches

It is important to understand that the classical approach of dual decomposition cannot be used for computing a schedule. The main reason is that the dual function

$$g(\lambda) = \max_{s \in \mathcal{S}} \{u(s) - \lambda^{\mathsf{T}} R s\} + \max_{c \in \mathcal{C}_{\text{STDMA}}} \{\lambda^{\mathsf{T}} c\}$$

is linear in c_l and will return a single transmission group in each iteration. Since each transmission group only activates a few links, many links will have zero rates until the next iteration of the algorithm can be carried out by the system and a new transmission group activated. If the communication overhead for solving the scheduling subproblem is negligible, it may be viable to perform the scheduling computations in each time slot. However, as we will see below, negotiating for transmission rights and adjusting transmission powers may require substantial overhead in interference-limited systems. It is then more attractive to maintain a transmission schedule with multiple time slots and update the schedule less frequently. As it turns out, a distributed algorithm for computing an asymptotically optimal schedule

can be derived either via mean value cross decomposition or primal decomposition combined with a conditional gradient method.

We will solve the following problem, which is a modified version of (4.3),

$$\begin{array}{ll} \underset{s,c}{\text{maximize}} & u(s) \\ \text{subject to} & Rs \leq c, \qquad s \in \mathcal{S} \\ & c \in \mathcal{C}_{\text{STDMA}}, \quad c_{\min} \leq c, \end{array} \tag{4.18}$$

where $\mathcal{S} = \{s | s_{\min} \leq s \leq s_{\max}\}$. For technical reasons[2], we add the constraints $s \leq s_{\max}$ and $c \geq c_{\min}$ to (4.3). If $c_{\min}$ is chosen sufficiently small, then the modified problem will have the same optimal solution as the original (4.3). In addition, simulations indicate that if $c_{\min}$ is small, it can in fact be neglected. We make the following assumption.

Assumption 4.7.1. *i)* $s_{\min} > 0$. *ii)* $c_{\min} > 0$. *iii)* $Rs_{\max} \geq c$ *for all* $c \in C_{\text{STDMA}}$. *iv) For all* $c \in \mathcal{C}_{\text{STDMA}}$, $c \geq c_{\min}$, *there is an* $s \in \mathcal{S}$ *such that* $Rs < c$.

These assumptions guarantee that the problem is feasible in s for all feasible c. Furthermore, they also guarantee that the optimal solution is limited by $\mathcal{C}_{\text{STDMA}}$ and not $s_{\max}$. Finally, they also imply that strong duality holds for the constraint $Rs \leq c$.

Mean Value Cross Decomposition Approach

First, we use a mean value cross decomposition approach to solve the modified problem (4.18). Recall that the mean value cross decomposition algorithm alternates between the primal and dual subproblems using the average values of the computed iterates as inputs; see Section 2.7.3. The primal subproblem

$$\begin{array}{ll} \underset{s}{\text{maximize}} & u(s) \\ \text{subject to} & Rs \leq \bar{c}^{(k)}, \quad s \in \mathcal{S}, \end{array} \tag{4.19}$$

with $\bar{c}^{(k)} = \frac{1}{k}\sum_{t=1}^{k} c^{(k)}$, gives the optimal Lagrange multipliers $\lambda^{(k)}$ for the capacity constraints. In addition, the (relevant part of the) dual subproblem

$$\begin{array}{ll} \underset{c}{\text{maximize}} & c^{\mathsf{T}}\bar{\lambda}^{(k)} \\ \text{subject to} & c \in \mathcal{C}_{\text{STDMA}}, \quad c_{\min} \leq c, \end{array} \tag{4.20}$$

[2]With the assumption that $u_p(s_p) \to -\infty$ when $s_p \to 0$ (to prevent starvation), we have the implicit condition that $s_p > 0$. This means that the feasible set is not closed and that the primal function cannot be differentiable on the whole set, $\mathcal{C}_{\text{STDMA}}$ (it is not even defined on the whole set). We need a closed feasible set and we will also need the primal function to be differentiable in one of our algorithms. Therefore, we require $s_p \geq s_{\min} > 0$, and in order to always have a feasible solution, we also require $c_l \geq c_{\min} > 0$ and $Rs_{\min} \leq c_{\min}$.

Algorithm 8 MVC

1: Let $k \leftarrow k_0$ and $\bar{c}^{(k_0)} > c_{\min}$.
2: **loop**
3: Solve (4.19) for $\bar{c}^{(k-1)}$ to obtain $\lambda^{(k)}$ and compute $\bar{\lambda}^{(k)}$.
4: Solve (4.20) for $\bar{\lambda}^{(k-1)}$ to obtain $c^{(k)}$.
5: Augment the schedule, compute $\bar{c}^{(k)}$, and let $k \leftarrow k+1$.
6: **end loop**

with $\bar{\lambda}^{(k)} = \frac{1}{k}\sum_{t=1}^{k} \lambda^{(k)}$, yields $c^{(k)}$. Since the primal subproblem is an instance of optimization flow control, mean value cross decomposition suggests the following approach for solving the network utility maximization problem (4.18): based on an initial schedule, we run the TCP/AQM scheme until convergence (this may require us to apply the schedule repeatedly). We refer to this phase as the data transmission phase. Nodes then record the associated equilibrium link prices for their transmitter queues and maintain their average values in memory. During the subsequent negotiation phase, which is described in detail in Section 4.7.2, we try to find the transmission group with largest average price-weighted throughput (solving (4.20)). We then augment the schedule with the corresponding link capacity vector (effectively increasing the number of time slots in the schedule by one). If the time slots are of equal length, the offered link capacities of the resulting schedule will equal the average of all computed transmission groups. The procedure is then repeated with the revised schedule. Our algorithm is summarized in Algorithm 8.

Theorem 4.7.2. *Under Assumptions 4.2.1 and 4.7.1, and that the scheduling subproblem can be solved to optimality, Algorithm 8 converges to the optimal solution to* (4.18)*, i.e.,*

$$\lim_{k\to\infty} u(\bar{s}^{(k)}) = u^\star \text{ and } \lim_{k\to\infty} \operatorname*{dist}_{C^\star}(\bar{c}^{(k)}) = 0,$$

with $\mathcal{C}^\star = \{c \in \mathcal{C}_{\text{STDMA}} | c_{\min} \le c, \max_{s\in\mathcal{S}}\{u(s)|Rs \le c\} = u^\star\}$.

Proof. The key idea is to identify the algorithm to be a mean value cross decomposition algorithm. We do the following identifications (with MVC notation to the left (where x is replaced with s) and STDMA algorithm notation on the right)

$$\begin{array}{llll} u(s) = u(s) & v(c) = 0 & A_1(s) = Rs & B_1(c) = -c \\ b_1 = 0 & A_2(s) = 0 & B_2(c) = 0 & b_2 = 0. \end{array}$$

The MVC primal subproblem (2.40) is identified with the STDMA primal subproblem (4.19). The MVC dual subproblem (2.39) is identified with the STDMA dual subproblem (4.20). Hence, the STDMA algorithm is an MVC algorithm and convergence follows from Holmberg and Kiwiel (2006); see Section 2.7.3. □

Note that an initial schedule can be constructed by letting $k_0 = \chi$ and using a pure TDMA schedule. Our theoretical analysis applies to the case when we augment the schedule indefinitely, while in practice one would like to use schedules with

limited frame length. Some suggestions for how the basic scheme can be adopted to this case can be found in Soldati et al. (2008). Although we do not have any theoretical results for computing schedules of fixed frame length, simulations reported in Soldati et al. (2008) indicate that the method can indeed be adopted to this case.

Conditional Gradient Approach

We also use a primal approach, more specifically a conditional gradient approach, to devise an algorithm for solving the modified problem (4.18). To prove convergence of the algorithm we need the following additional assumptions.

Assumption 4.7.3. *i) All links are bottlenecks. ii) The routing matrix has full row rank. iii) For any $c \in \mathcal{C}_{STDMA}$ with $c \geq c_{\min}$, we have that $s^\star(c) > s_{\min}$ in $\nu(c)$.*

The first assumption can be fulfilled by requiring that all links have at least one flow using only that link. The second assumption is a bit more restrictive compared to the assumptions needed for convergence of the MVC approach. But in the conditional gradient case, we do not need to use the average prices, which can lead to faster convergence.

We re-write the network utility maximization problem as

$$\begin{array}{ll} \underset{c}{\text{maximize}} & \nu(c) \\ \text{subject to} & c \in \mathcal{C}_{\text{STDMA}}, \quad c_{\min} \leq c, \end{array}$$

where we define the function $\nu : \mathbb{R}^\chi \to \mathbb{R}$ as,

$$\nu(c) = \max_{s \in \mathcal{S}} \big\{ u(s) \mid Rs \leq c \big\}. \tag{4.21}$$

For a fixed link capacity vector c, the function $\nu(c)$ can be evaluated via the optimization flow control algorithm, i.e., by letting the optimization flow control scheme converge. As will be shown shortly, under certain assumptions, $\nu(c)$ is differentiable with respect to c with derivative λ (the equilibrium link price vector for the network flow subproblem). Thus, in order to update the schedule and hence the link capacity vector c, it is natural to add the transmission group computed by the scheduling subproblem

$$\begin{array}{ll} \underset{c}{\text{maximize}} & c^\intercal \lambda^{(k)} \\ \text{subject to} & c \in \mathcal{C}_{\text{STDMA}}, \quad c_{\min} \leq c \end{array} \tag{4.22}$$

to the schedule. Effectively, this corresponds to a conditional gradient step in the ascent direction of $\nu(c)$; see Section 2.6.2. The augmented schedule is then applied to the system and the optimization flow control scheme, to evaluate (4.21), is run until convergence before the procedure is repeated. To describe the algorithm in detail, let $c^{(k)}$ be the transmission group computed in step k and let $\bar{c}^{(k)}$ denote

Algorithm 9 Conditional Gradient

1: Let $k \leftarrow k_0$ and $\bar{c}^{(k_0)} > 0$.
2: **loop**
3: Evaluate $\nu(\bar{c}^{(k)})$ by solving the optimization flow control problem (4.4), and let $\lambda^{(k)}$ be the associated equilibrium link prices.
4: Compute a new transmission group $c^{(k+1)}$ by solving the scheduling subproblem (4.23) for $\lambda^{(k)}$.
5: Augment the schedule with this transmission group and compute the associated $\bar{c}^{(k+1)}$.
6: Let $k \leftarrow k + 1$.
7: **end loop**

the average link-rate vector for a schedule consisting of k time-slots of equal length, i.e.,

$$\bar{c}^{(k)} = \frac{1}{k} \sum_{t=1}^{k} c^{(t)}.$$

For technical reasons, we added the constraint $c_{\min} \leq c$ to (4.18), but in this approach, we only need the constraint $c_{\min} \leq \bar{c}^{(k)}$, which is less conservative. The modified scheduling problem is

$$\begin{array}{ll} \underset{c}{\text{maximize}} & c^{\mathsf{T}} \lambda^{(k)} \\ \text{subject to} & c \in \mathcal{C}_{\text{STDMA}}, \quad c_{\min} \leq \frac{c + (k-1)\bar{c}^{(k-1)}}{k}, \end{array} \tag{4.23}$$

where $\bar{c}^{(k-1)}$ is fixed and the maximizer to this optimization problem is $c^{(k)}$ (note that $(c + (k-1)\bar{c}^{(k-1)})/k = \bar{c}^{(k)}$) . The problem (4.23) is almost the same as in the section on mean value cross decomposition, and an approximate solution method is presented in Section 4.7.2. Simulations indicate that the iterates $c^{(k)}$ do not tend to zero, and in practice, the constraint $\bar{c}^{(k)} \geq c_{\min}$ seems to be unnecessary and it can probably be neglected.

Our algorithm is summarized as Algorithm 9. As mentioned before, an initial schedule can, for example, be constructed by letting $k_0 = \chi$ and using a pure TDMA schedule.

This type of decomposition scheme falls into the class of *cross decomposition* methods (Holmberg, 1995) or conditional gradient methods (see Section 2.6.2). We will now proceed to show that it converges, but first, we consider some basic properties of $\nu(c)$ defined in (4.21).

Lemma 4.7.4. *Under Assumptions 4.2.1, 4.7.1, 4.7.3, the following holds.*

a) The function ν, defined in (4.21)*, is concave, and a subgradient to ν at c is given by the optimal Lagrange multiplier, $\lambda^\star(c)$, for the capacity constraint, $Rs \leq c$, in the definition of ν.*

b) The optimal Lagrange multiplier $\lambda^\star(c)$ corresponding to the capacity constraint, $Rs \leq c$, in the definition of ν, is unique.

c) The function $\nu(c)$ is continuously differentiable with $\frac{\partial}{\partial c}\nu(c) = \lambda^\star(c)$.

Proof. a) The concavity of ν follows from (Bertsekas et al., 2003, Proposition 6.5.1). Since (4.21) is the primal function of a feasible convex optimization problem, we have from Proposition 6.5.8 in Bertsekas et al. (2003) that $\lambda^\star(c)$ is an optimal Lagrange multiplier corresponding to the constraint $Rs \leq c$ in (4.21) if and only $\lambda^\star(c) \in \partial\nu(c)$.

b) The KKT conditions for the optimization problem (4.21) are necessary and sufficient, since it is a convex optimization problem with Slater's constraint qualification fulfilled. By assumption, we do not need to include the upper and lower limits on s. The KKT conditions are

$$\begin{aligned} u'(s^\star(c)) &= R^\intercal \lambda^\star(c) \\ Rs^\star(c) &= c. \end{aligned}$$

The latter expression holds by the assumption that all links are bottlenecks. The observation that the source rate $s^\star(c)$ is unique since the objective functions are strictly concave together with the fact that R has full row rank imply that $\lambda^\star(c)$ is unique.

c) Since all subgradients are Lagrange multipliers (from a)), which are unique for each c (from b)), the subdifferential contains only one element. Thus, ν is differentiable. To show that the derivative is continuous, we will use the implicit function theorem; see, e.g., Rudin (1976). For a given c, we get s and λ by setting $F(s, \lambda, c) = 0$ with

$$F(s, \lambda, c) = \begin{pmatrix} u'(s) - R^\intercal \lambda \\ Rs - c \end{pmatrix},$$

and we have

$$\begin{pmatrix} \nabla_s F(s, \lambda, c) \\ \nabla_\lambda F(s, \lambda, c) \end{pmatrix} = \underbrace{\begin{pmatrix} \operatorname{diag}\big(u_1''(s_1) & \dots & u_\eta''(s_\eta)\big) & R^\intercal \\ -R & & 0 \end{pmatrix}}_{A(c)},$$

where $A(c)$ is invertible. To see this, assume there exists $\begin{pmatrix} x^\intercal & y^\intercal \end{pmatrix}^\intercal \neq 0$ such that $A(c)\begin{pmatrix} x^\intercal & y^\intercal \end{pmatrix}^\intercal = 0$, which implies that

$$x = -\operatorname{diag}\big(u_1''(s_1) \quad \dots \quad u_\eta''(s_\eta)\big)^{-1} R^\intercal y$$

and

$$-R\,\mathrm{diag}\big(u_1''(s_1) \quad \dots \quad u_\eta''(s_\eta)\big)^{-1} R^\intercal y = 0.$$

Furthermore, we have that

$$(y^\intercal R)\,\mathrm{diag}\big(u_1''(s_1) \quad \dots \quad u_\eta''(s_\eta)\big)^{-1} (R^\intercal y) = 0,$$

which can only be fulfilled with $R^\intercal y = 0$ since $\mathrm{diag}\big(u_1''(s_1) \quad \dots \quad u_\eta''\big)$ is non-singular. Since R is assumed to be full row rank, $R^\intercal y = 0$ is impossible with $x \neq 0$ and $y \neq 0$. Hence, $A(c)$ is invertible. The implicit function theorem now gives us that $\begin{pmatrix} s^\intercal & \lambda^\intercal \end{pmatrix}^\intercal = \varphi(c)$ and

$$\nabla\varphi(c) = -\nabla_c F(s, \lambda, c)\,(A(c))^{-1} = \begin{pmatrix} 0 & I \end{pmatrix} (A(c))^{-1}.$$

Since λ is differentiable with respect to c, λ is also continuous with respect to c.

□

We are now ready for the main convergence theorem.

Theorem 4.7.5. *Let $u^\star$ be the optimal value of the centralized cross-layer design problem* (4.18)*. Under Assumptions 4.2.1, 4.7.1, and 4.7.3, and that the scheduling subproblem can be solved to optimality, Algorithm 9 converges in the sense that*

$$\lim_{k\to\infty} \nu(\bar{c}^{(k)}) = u^\star.$$

Proof. The update rule for $\bar{c}^{(k)}$ can be re-written as

$$\bar{c}^{(k+1)} = \frac{k}{k+1}\bar{c}^{(k)} + \frac{1}{k+1}c' = (1-\omega_k)\bar{c}^{(k)} + \omega_k c',$$

where $\omega_k = 1/(k+1)$ and c' is found as the solution to the congestion-weighted throughput problem

$$\begin{aligned} \text{maximize} \quad & \lambda^T c' \\ \text{subject to} \quad & c' \in \mathcal{C} \\ & \bar{c}^{(k)} \geq c_{\min}. \end{aligned}$$

Since λ is a gradient of $\nu(\bar{c}^{(k)})$ at $\bar{c}^{(k)}$, this is a conditional gradient method with an open loop stepsize rule (Dunn and Harshbarger, 1978), see Section 2.6.2, that satisfies

$$\omega^{(k+1)} = \frac{\omega_k}{\omega_k + 1}, \quad \omega_0 = 1.$$

Since $\lambda^\star$ is continuous and the domain is compact (since $\bar{c}^{(k)} \in \mathcal{C}$ and $\bar{c}^{(k)} \geq c_{\min}$), the derivative of $\nu(\bar{c}^{(k)})$ is uniformly continuous. By Theorem 1 in Dunn and Harshbarger (1978), we have that $\lim_{k\to\infty} \nu(\bar{c}^{(k)}) = u^\star$. Thus, Algorithm 9 converges to the optimal solution. □

As stated before, the lower limit, $c_{\min}$, on $\bar{c}^{(k)}$ does not seem to be necessary. We have the following conjecture:

Conjecture 4.7.6. *Under Assumptions 4.2.1, 4.7.1, and 4.7.3, and that the scheduling subproblem can be solved to optimality, but without the explicit enforcement of $\bar{c}^{(k)} \geq c_{\min}$, then Algorithm 9 converges to the optimal solution to* (4.18)*, i.e.,* $\lim_{k\to\infty} \nu(\bar{c}^{(k)}) = u^\star$.

4.7.2 Solving the Scheduling Subproblem

The final component of a distributed solution to the NUM problem for STDMA networks is a distributed mechanism for solving the scheduling subproblems (4.20) and (4.23) in each negotiation phase. We will neglect the constraints $c \geq c_{\min}$ and $\bar{c}^{(k)} \geq c_{\min}$, and the subproblems are therefore identical except that the MVC algorithm uses the average link prices and the cross decomposition algorithm uses the instantaneous link prices.

Although a fully distributed scheme that solves the subproblem to optimality appears out of reach, a suite of suboptimal schemes have been proposed and investigated in Soldati et al. (2008). We will outline one of these approaches below.

Since the scheduling subproblems (4.20) and (4.23) are linear in c_l, an optimal solution can always be found at a vertex of the capacity region, i.e., among one of the feasible transmission groups. We will consider a distributed solution that is based on two logical steps: first, a candidate transmission group is formed by trying to maximize the objective function subject to primary interference constraints only; then, transmitters adjust their powers to allow the most advantageous subset of candidate links to satisfy the secondary interference constraints. Clearly, some links may need to drop out of the candidate set during the power negotiation phase, and the resulting transmission group may be suboptimal.

The candidate group formation is based on the observation that the primary constraints are satisfied if only one link in each two-hop neighborhood is activated. In an attempt to maximize the objective of the dual subproblem, the link with the highest average link price (MVC case) or instantaneous link price (cross decomposition case) in a two-hop neighborhood will assign itself membership to the candidate set. To allow links to make this decision, we assume that the transmitters of each link forwards information about its link price to the receiving node. By collecting the maximum link prices from its neighbors, each node can decide if one of its own transmitters should enter the candidate set or remain silent.

Once a link has identified itself as a member of the candidate set, it will start contending for transmission rights. In our approach, links enter the transmission group one-by-one, adjusting their transmit powers to maximize the number of links in the

transmission group. The approach exploits the properties of distributed power control with active link protection (DPC/ALP) (Bambos et al., 2000). The DPC/ALP algorithm is an extension of the classical distributed power control algorithms (e.g., Foschini and Miljanic (1993)) which maintains the quality of service of operational links (link protection) while allowing inactive links to gradually power up in order to try to enter the transmission group. As interference builds up, active links sustain their quality while new ones may be blocked and denied channel access. The DPC/ALP algorithm exploits local measurements of SINRs at the receivers and runs iteratively. To describe the algorithm, we introduce $\mathcal{A}$ and $\mathcal{I}$ as the set of active and inactive links, and let γ_l be the measured SINR on link l. The DPC/ALP algorithm operates by updating the transmit powers P_l according to

$$P_l^+ = \begin{cases} \delta P_l \gamma^{\mathrm{tgt}}/\gamma_l & \text{if } l \in \mathcal{A} \\ \delta P_l & \text{if } l \in \mathcal{I}, \end{cases} \tag{4.24}$$

where $\delta > 1$ is a control parameter and P_l^+ denotes the next iterate of P_l. Links change status from inactive to active when their measured SINR exceeds the target. Inactive nodes that consistently fail to observe any SINR improvement enters a voluntary drop-out phase and go silent (see Bambos et al. (2000) for details).

The negotiation phase is initialized by letting $\mathcal{I}$ equal the candidate set and letting $\mathcal{A}$ be empty. Links then execute the DPC/ALP algorithm. To increase the likelihood of forming a transmission group with high price-weighted throughput, we let links wait a random time before starting to execute the DPC/ALP algorithm. The waiting probability is a decreasing function of the link price, so that highly loaded links will tend to start ramping up transmit powers before lightly loaded ones (and thus, have increased chances to obtain transmission rights). At the end of the negotiation phase, $\mathcal{A}$ constitutes a feasible transmission group.

4.7.3 Numerical Results

We now demonstrate the typical performance of Algorithm 8 (MVC) and Algorithm 9 (conditional gradient algorithm or cross decomposition algorithm) by applying them to the hypothetical indoor wireless LAN scenario described in Johansson and Xiao (2006). The network topology with 10 nodes and 36 links can be seen in Fig. 4.12. Figure 4.13 shows the objective value versus schedule length. The MVC and conditional gradient algorithms start with an initial schedule consisting of an equal time slot length TDMA schedule. The solid line is the optimal STDMA performance computed using the off-line approach described in Johansson and Xiao (2006), the dash dotted line is the MVC algorithm with the subproblem solved to optimality, and finally, the dashed line is the cross decomposition algorithm with the subproblem solved to optimality. The performance of the optimal TDMA schedule is not shown in the figure, but the objective value of this scheme is circa 53. Thus, Algorithm 8 and Algorithm 9 outperform the optimal (variable time slot length) TDMA schedule, and they tend to the optimal value. Figure 4.14 shows

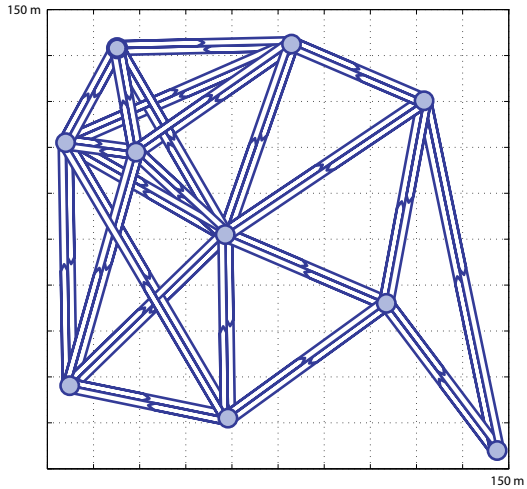


Figure 4.12: The node placement of the example network. The filled circles denote the nodes and the lines indicate the links in the network.

the network utility as a function of the number of iterations for three versions of the cross decomposition algorithm. The dashed line denotes the cross decomposition algorithm, where a slot is added at each iteration (the number of time slots goes towards infinity), i.e., Algorithm 9. The two other lines denote the rolling horizon case: a fixed number time slots is used where the oldest transmission group is replaced by the most recently generated transmission group. Note that the scheduling subproblem is solved to optimality in all the simulations above. There will be some performance loss when a suboptimal (heuristic) procedure is used to solve the scheduling subproblem. However, Soldati et al. (2008) show simulations for a number of heuristical methods for solving the scheduling subproblem, and the performance loss is not that great. The details on the performance loss as well as more comparative simulations can be found in Soldati et al. (2008).

4.8 Summary

This chapter has used the decomposition techniques presented in Chapter 2 to design utility-maximizing mechanisms for two networking technologies.

We have presented a flow chart that can be useful in order to categorize and visualize previous results in the NUM literature. The flowchart highlights the inherent steps in posing, decomposing, and solving NUM optimization problems. In addition, the flowchart, with examples and a tutorial, is available on-line on the Internet. This opens up for the possibility of refining the flowchart and adding relevant results when they appear.

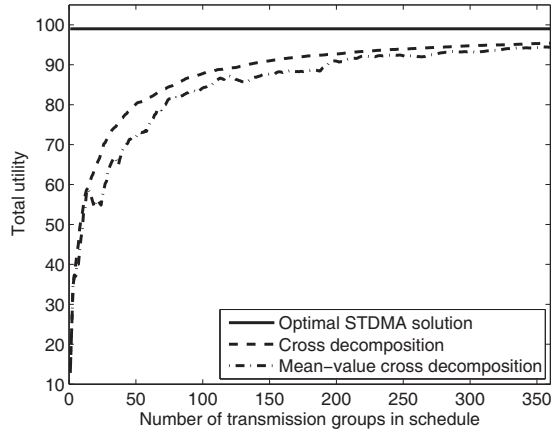


Figure 4.13: Network utility as a function of schedule length for a number of alternative schemes. The optimal TDMA schedule (with variable slot length) yields a utility of circa 53 (not shown in the figure).

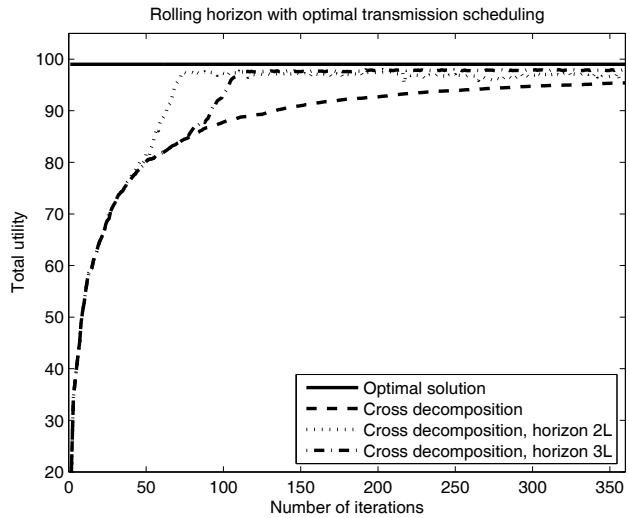


Figure 4.14: Network utility as a function of the number of iterations for three versions of the cross decomposition algorithm are shown. The dashed line denotes the cross decomposition where a slot is added at each iteration (the number of time slots grows at the same pace as the number of iterations). The two other lines denote the rolling horizon case: a fixed number time slots is used where the oldest transmission group is replaced by the most recently generated transmission group.

We developed two center-free algorithms that optimally allocate the spectrum in a FDMA network. The first algorithm is based in dual relaxation and the second algorithm is based in primal decomposition. The algorithms have different time-scale properties.

Next, an algorithm to be used to devise a schedule in a STDMA network was developed. The algorithm is based on premise that the schedule is augmented indefinitely, and we can show that the algorithm asymptotically reaches the optimum under certain assumptions. However, an infinite schedule is not practical, and in practice, the schedule length is fixed. The most natural way of addressing this potential problem is to remove the oldest slot and replace it with a new slot. It turns out that this heuristic works very well in simulations. Nevertheless, the performance loss should be quantified and also bounded if possible.

Although the theory for network utility maximization is evolving quickly, much work remains to be done. This includes developing better tools for analyzing protocol dynamics to guarantee stable and efficient system behavior, and improved support for understanding the dependencies that the protocols introduce between networking layers. On the practical side, complete implementations of the NUM based protocols should be developed and their performance should be assessed in network simulators under realistic traffic and radio models.

Chapter 5

Optimal Consensus

"*It is perhaps natural that the concept of best or optimal decisions should emerge as the fundamental approach for formulating decision problems.*"

D. G. Luenberger, Optimization by Vector Space Methods, 1969

The meaning of optimal must of course be carefully defined, especially in a thesis on optimization. In this chapter, we consider some variations of the canonical problem of consensus, already touched upon in Section 2.3. First, in Section 5.1, we consider the rendezvous problem of several dynamical agents, or subsystems, described by general linear state equations in discrete-time. The novelty lies in that instead of studying convergence to the barycenter of the initial states, we devise an algorithm where the agents find the optimal rendezvous point, in the sense that the agents' trajectories to the rendezvous point minimize a quadratic cost function (the exact definition of optimality for this case is given in (5.4)). Second, in Section 5.2, we study techniques for accelerating linear iterations that converge to the barycenter of the initial state (the basic algorithm can be found in Section 2.3). Since rapid convergence of the consensus iterations is crucial for the performance of some of the algorithms in Chapter 3, this is a significant problem.

The detailed outline of the chapter is as follows. In Section 5.1, we consider the rendezvous problem of several dynamical agents. We start with some background material in Section 5.1.1. Then, in Section 5.1.2 we define the specific problem, optimal consensus, and in Section 5.1.3, we develop a distributed negotiation scheme that computes the optimal consensus point. Section 5.1.4 contains numerical experiments that illustrate the developed method. In Section 5.2, we study techniques for accelerating linear iterations that converge to the barycenter of the initial state. More specifically, we investigate the benefits of algorithms with an augmented state vector. Section 5.2.1 outlines related work and Section 5.2.2 introduces the so-called the shift-register case. Then, in Section 5.2.3, we discuss how the convergence of the linear iteration can be accelerated by using different off-line optimization schemes. Next, in Section 5.2.4, we quantify the benefits of these schemes using numerical

examples. In Section 5.2.5, we provide necessary and sufficient conditions for convergence of a more general case of the linear iterations. Finally, we conclude the chapter with a summary in Section 5.3.

5.1 Optimal Multi-Agent Consensus

The problem of cooperatively controlling systems composed of a large number of autonomous agents has attracted substantial attention in the control and robotics communities. An interesting instantiation is the consensus problem. It consists of designing distributed control strategies such that the state or output of a group of agents asymptotically converges to a common value, a *consensus point*. The agents are typically modeled by identical first-order linear systems with no state nor input constraints.

The main contribution of this section is a decentralized negotiation algorithm that computes the optimal consensus point for a set of agents modeled as linear control systems. The consensus point is a vector that specifies, for example, the position and velocity the agents shall converge to. Our approach allows us to incorporate constraints on the state and the input, which is not easily done for the traditional consensus algorithm; see the discussion in Marden et al. (2007). By primal decomposition and incremental subgradient methods we design a decentralized negotiation algorithm, in which each agent performs individual planning of its trajectory and exchanges only a small amount of information per iteration with its neighbors. We show that the cost of reaching the consensus point can be significantly reduced, by letting the agents negotiate to find an optimal or near optimal consensus point, before applying a control signal.

5.1.1 Background

There has been a lot of research activity in this area, and a good starting point for related work is the recent survey paper Olfati-Saber et al. (2007). In particular, if the consensus point is a position and *fixed a priori*[1] (contrary to our approach, where the optimal consensus point is a decision variable) we get a so-called rendezvous problem. For this type of problem, much work have been focused on establishing convergence to the fixed consensus point under different communication and visibility conditions; see for example Cortéz et al. (2006) and the references therein. Furthermore, optimal control formulations have been used in papers that focus on the convergence of distributed model predictive control (MPC) based strategies to an *a priori* fixed equilibrium point. Dunbar and Murray (2006) propose a decentralized scheme where a given desired equilibrium point is asymptotically reached. The scheme requires coupled subsystems to update and exchange the most recent

[1] In the consensus literature, the consensus point is typically fixed in the sense that it is computed from the initial conditions using a simple rule, for example, the consensus point could be the average of the starting positions of the agents.

optimal control trajectories prior to each update step. Stability is guaranteed if each subsystem does not deviate too far from the previous open-loop trajectory. In Keviczky et al. (2006), the authors propose a strategy where each subsystem solves a finite time optimal control problem. The solution of the problem requires each subsystem to know the neighbors' model, constraints, and state. The strategy also requires the prior knowledge of an overall system equilibrium. Finally, a related distributed optimization problem, focused on formation flight, is considered in Raffard et al. (2004), where the decentralized algorithm is based on dual relaxation. Their approach differs from ours in that they do not consider the consensus problem and that they use dual relaxation instead of primal decomposition.

5.1.2 Problem Formulation

Consider $n > 1$ agents whose dynamics are described by

$$\begin{aligned} x_i(t+1) &= A_i x_i(t) + B_i u_i(t) \\ z_i(t) &= C_i x_i(t)\,, \qquad\qquad i = 1, \ldots, n\,, \end{aligned} \tag{5.1}$$

where $A_i \in \mathbb{R}^{n_i \times n_i}$, $B_i \in \mathbb{R}^{n_i \times p_i}$, and $C_i \in \mathbb{R}^{\xi \times n_i}$ are observable and controllable. The vector $x_i(0) = x_i^0 \in \mathbb{R}^{n_i}$ is the initial condition and $z_i(t)$ is the performance output. We assume that the inputs are constrained according to

$$\left(u_i^\intercal(0), u_i^\intercal(1), \ldots, u_i^\intercal(\tau)\right)^\intercal \in \mathcal{U}_i,\ i = 1, \ldots, n, \tag{5.2}$$

where τ is a (fixed) time horizon and $\mathcal{U}_i$ is a convex set. By using standard techniques from MPC, the constraint can encode magnitude and rate constraints on $u_i(t)$, as well as restrictions on linear combinations of the agent states (Maciejowski, 2002, Sec 3.2).

Definition 5.1.1. *Let θ lie in a compact, closed, and convex set $\Theta \subset \mathbb{R}^\xi$. The agents described by (1) reach* consensus[2] *at time τ if*

$$z_i(\tau + k) = \theta,\ \textit{for all } k \geq 0 \textit{ and } i = 1, \ldots, n,$$

with $u_i(\tau + k) = u_i(\tau)$, for all $k \geq 0$ and $i = 1, \ldots, n$.

The objective is to find a consensus point $\theta \in \Theta$ and a sequence of inputs $\left(u_i^\intercal(0), u_i^\intercal(1), \ldots, u_i^\intercal(\tau)\right)^\intercal \in \mathcal{U}_i$, with $i = 1, \ldots, n$, such that consensus is reached at time τ. The following cost function is associated to the ith system:

$$\begin{aligned} V_i(z_i(t), u_i(t-1), \theta) \triangleq\ & (z_i(t) - \theta)^\intercal Q_i\, (z_i(t) - \theta) \\ & + u_i(t-1)^\intercal R_i u_i(t-1), \end{aligned} \tag{5.3}$$

[2]By introducing a fixed offset, $\bar{\theta}_i$, one for each agent, it is possible to define a *consensus formation* relative to a global consensus point θ. The condition of consensus formation is that $z_i(\tau + k) = \theta + \bar{\theta}_i$, for all $k \geq 0$ and $i = 1, \ldots, n$.

where $Q_i \in \mathbb{R}^{\xi\times\xi}$ and $R_i \in \mathbb{R}^{p_i\times p_i}$ are positive definite symmetric matrices that encode the cost of deviating from the consensus point and the cost of control energy for agent i. Let us introduce the following vectors:

$$\mathbf{x}_i \triangleq \left(x_i^\intercal(1), x_i^\intercal(2), \dots, x_i^\intercal(\tau+1)\right)^\intercal$$
$$\mathbf{u}_i \triangleq \left(u_i^\intercal(0), u_i^\intercal(1), \dots, u_i^\intercal(\tau)\right)^\intercal.$$

Since

$$\mathbf{x}_i = \underbrace{\begin{pmatrix} A_i \\ A_i^2 \\ \vdots \\ A_i^{\tau+1} \end{pmatrix}}_{\mathbf{E}_i} x_i^0 + \underbrace{\begin{pmatrix} B_i & 0 & \dots & 0 \\ A_iB_i & B_i & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A_i^\tau B_i & A_i^{\tau-1}B_i & \dots & B_i \end{pmatrix}}_{\mathbf{F}_i} \mathbf{u}_i\,,$$

we have $z_i(\tau) = C_i x_i(\tau) = \mathbf{H}_i(\mathbf{E}_i x_i^0 + \mathbf{F}_i\mathbf{u}_i) = \theta$, where $\mathbf{H}_i \triangleq \begin{pmatrix} 0 & \dots & C_i & 0 \end{pmatrix}$. We also introduce $\mathbf{U}_i \triangleq A_i^{\tau+1} - A_i^\tau$ and $\mathbf{W}_i \triangleq \begin{pmatrix} A_i^\tau B_i & A_i^{\tau-1}B_i & \dots & B_i \end{pmatrix} - \begin{pmatrix} A_i^{\tau-1}B_i & A_i^{\tau-2}B_i & \dots & 0 \end{pmatrix}$. We now formulate the optimization problem,

$$\underset{\mathbf{u}_1,\dots,\mathbf{u_n},\,\theta}{\text{minimize}} \quad \sum_{i=1}^{n} \mathbf{V}_i(\mathbf{u}_i, \theta) \tag{5.4a}$$
$$\text{subject to} \quad \mathbf{H}_i(\mathbf{E}_i x_i^0 + \mathbf{F}_i\mathbf{u}_i) = \theta,\ i = 1,\dots,n \tag{5.4b}$$
$$\mathbf{U}_i x_i^0 + \mathbf{W}_i\mathbf{u}_i = 0,\ i = 1,\dots,n \tag{5.4c}$$
$$\mathbf{u}_i \in \mathcal{U}_i,\ i = 1,\dots,n \tag{5.4d}$$
$$\theta \in \Theta, \tag{5.4e}$$

with the cost function

$$\begin{aligned}\mathbf{V}_i(\mathbf{u}_i, \theta) &\triangleq \sum_{t=1}^{\tau+1} V_i(z_i(t), u_i(t-1), \theta) \\ &= (\mathbf{C}_i(\mathbf{E}_i x_i^0 + \mathbf{F}_i\mathbf{u}_i) - \mathbf{1}_{\tau+1}\otimes\theta)^\intercal \mathbf{Q}_i(\mathbf{C}_i(\mathbf{E}_i x_i^0 \\ &\quad + \mathbf{F}_i\mathbf{u}_i) - \mathbf{1}_{\tau+1}\otimes\theta) + \mathbf{u}_i^\intercal\mathbf{R}_i\mathbf{u}_i,\end{aligned}$$

where[3] $\mathbf{Q}_i = I_{\tau+1}\otimes Q_i$, $\mathbf{R}_i = I_{\tau+1}\otimes R_i$, and $\mathbf{C}_i = I_{\tau+1}\otimes C_i$. Notice that the constraint (5.4b) guarantees consensus at time τ and (5.4c) guarantees that the consensus point is an equilibrium, i.e., $x_i(\tau) = A_i x_i(\tau) + B_i u_i(\tau)$. The constraint (5.4b) can potentially lead to infeasibility problems, but such problems can be mitigated by replacing the constraint with a penalty term in the objective, penalizing deviations

[3] With $\mathbf{1}_{\tau+1}$ we denote the column vector with $\tau+1$ ones, with $I_{\tau+1}$ the $\tau+1\times\tau+1$ identity matrix, and with $\otimes$ the Kronecker matrix product.

from the consensus point at time τ. Note, however, that due to Assumption 5.1.3 infeasibility problems do not arise in our setup.

We make the following standing assumptions, which make the optimization problem (5.4) convex and feasible, and guarantee that the consensus point is an equilibrium.

Assumption 5.1.2. *The matrices $Q_i \in \mathbb{R}^{\xi\times\xi}$ and $R_i \in \mathbb{R}^{p_i\times p_i}$, $i = 1,\dots,n$, are positive definite and symmetric. The set Θ is convex, closed, and compact. The sets $\mathcal{U}_i$, $i = 1,\dots,n$, are convex and closed.*

Assumption 5.1.3. *For all $\theta \in \Theta$, $x_i \in \{y \in \mathbb{R}^{n_i} | C_i y = \theta\}$, and $i = 1,\dots,n$, there exists $\mathbf{u}_i$ in the relative interior of $\mathcal{U}_i$ such that $z_i(\tau) = \theta$ and $x_i = A_i x_i + B_i u_i(\tau)$.*

The optimization problem (5.4) is interesting for a multi-agent setting if the computations can be distributed among the agents and the amount of information that they need to exchange is limited. In the following we develop a negotiation algorithm to find the optimal consensus point, in which agents exchange only their current estimates of θ.

5.1.3 Distributed Negotiation

To distribute the computation of the optimal consensus point, we use primal decomposition in combination with an incremental subgradient method; see Section 2.6.1 and Section 3.3. Let us start with defining $\nu_i(\theta)$ as follows

$$\begin{aligned} \nu_i(\theta) = \min_{\mathbf{u}_i} \quad & \widehat{\mathbf{V}}_i(\mathbf{u}_i) \\ \text{subject to} \quad & \mathbf{H}_i(\mathbf{E}_i x_i^0 + \mathbf{F}_i \mathbf{u}_i) = \theta \\ & \mathbf{U}_i x_i^0 + \mathbf{W}_i \mathbf{u}_i = 0 \\ & \mathbf{u}_i \in \mathcal{U}_i, \end{aligned} \tag{5.5}$$

where $\widehat{\mathbf{V}}_i(\mathbf{u}_i) \triangleq \mathbf{V}_i(\mathbf{u}_i, \theta)$; we have eliminated the dependence on θ in $\widehat{\mathbf{V}}_i(\mathbf{u}_i)$ by using the constraint $\mathbf{H}_i(\mathbf{E}_i x_i^0 + \mathbf{F}_i \mathbf{u}_i) = \theta$. Also note that we can re-write (5.5) as

$$\nu_i(\theta) = \min_{\mathbf{u}_i} \left\{ \mathbf{u}_i^\intercal \Xi \mathbf{u}_i + \Psi \mathbf{u}_i + r \,\middle|\, \Gamma \mathbf{u}_i = b,\, \mathbf{u}_i \in \mathcal{U}_i \right\},$$

with

$$\begin{aligned}
\Xi &= (\mathbf{C}_i\mathbf{F}_i)^\intercal \mathbf{Q}_i \mathbf{C}_i\mathbf{F}_i + \mathbf{R}_i \\
\Psi &= 2(\mathbf{C}_i\mathbf{E}_i x_i^0)^\intercal \mathbf{Q}_i \mathbf{C}_i\mathbf{F}_i - 2(\mathbf{1}_{\tau+1} \otimes \theta)^\intercal \mathbf{Q}_i \mathbf{C}_i\mathbf{F}_i \\
r &= (\mathbf{C}_i\mathbf{E}_i x_i^0)^\intercal \mathbf{Q}_i \mathbf{C}_i\mathbf{E}_i x_i^0 + (\mathbf{1}_{\tau+1} \otimes \theta)^\intercal \mathbf{Q}_i \mathbf{1}_{\tau+1} \otimes \theta \\
\Gamma &= \begin{pmatrix} \mathbf{H}_i\mathbf{F}_i \\ \mathbf{W}_i \end{pmatrix} \\
b &= \begin{pmatrix} \theta - \mathbf{H}_i\mathbf{E}_i x_i^0 \\ -\mathbf{U}_i x_i^0 \end{pmatrix}.
\end{aligned}$$

Thus, (5.5) is a quadratic program with linear and convex constraints. Furthermore, note that the optimization problem (5.4) can be written as

$$\begin{aligned}
&\underset{\theta}{\text{minimize}} && \sum_{i=1}^{n} \nu_i(\theta) \\
&\text{subject to} && \theta \in \Theta\,.
\end{aligned} \tag{5.6}$$

We then have the following result.

Lemma 5.1.4. *The cost function ν_i defined in* (5.5) *is a convex function. A subgradient λ_i for ν_i at θ is given by the Lagrange multipliers corresponding to the constraint $\mathbf{H}_i(\mathbf{E}_i x_i^0 + \mathbf{F}_i\mathbf{u}_i) = \theta$.*

Proof. By Lagrangian relaxation we can define

$$\mathrm{L}_i(\mathbf{u}_i, \theta, \lambda_i) = \widehat{\mathbf{V}}_i(\mathbf{u}_i) - \lambda_i^\intercal(\mathbf{H}_i(\mathbf{E}_i x_i^0 + \mathbf{F}_i\mathbf{u}_i) - \theta)\,,$$

where λ_i are Lagrange multipliers. We also introduce the dual function

$$\mathrm{d}_i(\lambda_i, \theta) = \min_{\mathbf{u}_i \in \tilde{\mathcal{U}}_i} \left\{ \widehat{\mathbf{V}}_i(\mathbf{u}_i) - \lambda_i^\intercal(\mathbf{H}_i(\mathbf{E}_i x_i^0 + \mathbf{F}_i\mathbf{u}_i) - \theta) \right\},$$

where $\tilde{\mathcal{U}}_i = \{u_i \in \mathcal{U}_i | \mathbf{U}_i x_i^0 + \mathbf{W}_i\mathbf{u}_i = 0\}$. Strong duality follows from Lemma 2.7.1, since

1. the constraint $\mathbf{H}_i(\mathbf{E}_i x_i^0 + \mathbf{F}_i\mathbf{u}_i) = \theta$ is linear in $\mathbf{u}_i$,

2. Assumption 5.1.3 guarantees that there exists a solution in the relative interior of $\mathcal{U}_i$ to this equation,

3. the function $\widehat{\mathbf{V}}_i$ and the set $\mathcal{U}_i$ are convex.

Hence, $\nu_i(\theta) = \max_{\lambda_i} \mathrm{d}_i(\lambda_i, \theta)$. Consider two feasible points, $\theta^{\downarrow}$ and $\theta^{\ddagger}$, and let $\lambda_i^{\downarrow}$ be the Lagrange multipliers corresponding to the relaxed constraint for $\theta^{\downarrow}$, then

$$\begin{aligned}\nu_i(\theta^{\ddagger}) &= \max_{\lambda_i} \mathrm{d}_i(\lambda_i, \theta^{\ddagger}) \geq \mathrm{d}_i(\lambda_i^{\downarrow}, \theta^{\downarrow}) + (\lambda_i^{\downarrow})^{\intercal}(\theta^{\ddagger} - \theta^{\downarrow}) \\ &= \nu_i(\theta^{\downarrow}) + (\lambda_i^{\downarrow})^{\intercal}(\theta^{\ddagger} - \theta^{\downarrow}).\end{aligned}$$

Hence, by the definition of a subgradient, $\lambda_i^{\downarrow}$ is a subgradient of ν_i at $\theta^{\downarrow}$. Now $\nu_i(\theta^{\ddagger})$ can be expressed as

$$\begin{aligned}\nu_i(\theta^{\ddagger}) &= \max_{\lambda_i} \left\{ \mathrm{d}_i(\lambda_i, \theta) + \lambda_i^{\intercal}(\theta^{\ddagger} - \theta) \right\} \\ &= \max_{\lambda_i} \left\{ g(\lambda_i) + \lambda_i^{\intercal}\theta^{\ddagger} \right\},\end{aligned}$$

where $g(\lambda_i) = \mathrm{d}_i(\lambda_i, \theta) - \lambda_i^{\intercal}\theta$ and $g(\lambda_i) + \lambda_i^{\intercal}\theta^{\ddagger}$ is affine in $\theta^{\ddagger}$. Since $\nu_i(\theta^{\ddagger})$ is the pointwise maximum of a family of convex functions, $\nu_i(\theta^{\ddagger})$ is convex. □

The optimal consensus point can be computed in a distributed way using the incremental subgradient methods from optimization theory; see Section 2.6.1. In this scheme, an estimate of the optimal consensus point is passed between agents. Upon receiving an estimate from its neighbor, an agent solves the optimization problem (5.5) to evaluate its cost of reaching the suggested consensus point and to compute an associated subgradient (using Lemma 5.1.4). The agent then updates the consensus estimate via

$$\theta^{(k+1)} = \mathcal{P}_{\Theta}[\theta^{(k)} - \alpha_h \lambda_{i,k}] \tag{5.7}$$

and passes the estimate to the next agent. The algorithm proceeds iteratively. Here $\mathcal{P}_{\Theta}[\cdot]$ denotes the Euclidean projection on the set Θ, α_h is the stepsize, and $\lambda_{i,k}$ is a subgradient of ν_i at $\theta^{(k)}$. In addition, pseudocode of the algorithm is given in Algorithm 10. As we have seen, the difference between the incremental subgradient method and the vanilla subgradient method is that each agent only computes a subgradient with respect to its own part of the objective function and not the global objective function. Furthermore, the convergence of the incremental subgradient algorithm is guaranteed if the agents can be organized into a cycle graph, which we formalize in the following assumption.

Assumption 5.1.5. *The agents can be organized into a cycle graph and neighboring nodes in this graph can communicate with each other.*

The following lemma guarantees convergence.

Lemma 5.1.6. *Under the Assumptions 5.1.2, 5.1.3, and 5.1.5, Algorithm 10 converges to an optimizer of problem* (5.4).

Algorithm 10 Cyclic Incremental Algorithm.

Initialize θ_0 and α_0. Set $k \leftarrow 0$ and $h \leftarrow 1$.
loop
  $\alpha_h \leftarrow \alpha_0/h$
  for $i = 1$ to n **do**
    Compute a subgradient, $\lambda_{i,k}$, for $\nu_i(\theta^{(k)})$
    $\theta^{(k+1)} \leftarrow \mathcal{P}_\Theta[\theta^{(k)} - \alpha_h \lambda_{i,k}]$
    $k \leftarrow k + 1$
  end for
  $h \leftarrow h + 1$
end loop

Proof. The proof follows from Theorem 8.2.6 (p. 480) and Theorem 8.2.13 (p. 496) in Bertsekas et al. (2003), since the set Θ is convex, closed, and compact (so the norms of all subgradients have an upper bound and the projection can be carried out), and the stepsize α_h is square summable over h, but not summable over h. □

Algorithm 10 can be modified to a randomized version as described in Chapter 3, which corresponds to that the estimate is sent to a random agent at each update. Regardless if the deterministic or the randomized version of the algorithm is used, the convergence behavior is asymptotic, which means that some stopping criteria need to be used. The simplest and most practical criteria is to negotiate for a fixed number of iterations. More advanced stopping criteria are of course possible, but these are outside the scope of this thesis.

5.1.4 Control Strategies and Numerical Examples

In this section we discuss control strategies and possible extensions. The simplest way to devise a control strategy from the problem formulation is to first execute a *negotiation phase* in which Algorithm 10 is run until a sufficiently accurate optimal consensus point is agreed upon and then, in a *motion phase*, apply the corresponding open-loop control to reach it. If there are no disturbances the system will reach the consensus point at time τ. The main advantage of the proposed strategy is that the optimal consensus point is computed in a distributed way and only a small amount of information, the current consensus point estimate, needs to be exchanged at each step. To make the strategy robust to disturbances, the motion phase can be performed using closed-loop feedback control with the optimal consensus point as a set point. The controller could be devised using, for example, MPC techniques.

We explore the performance of the distributed negotiation. The setup is as follows: three agents with double integrator dynamics and input constraints ($|u_i| \leq 1$) should reach the same coordinates at time $\tau = 40$. The convergence rate of the consensus point negotiation is shown in Figure 5.1. The iteration can clearly be seen to converge, and the convergence rate is high in the beginning but slows down

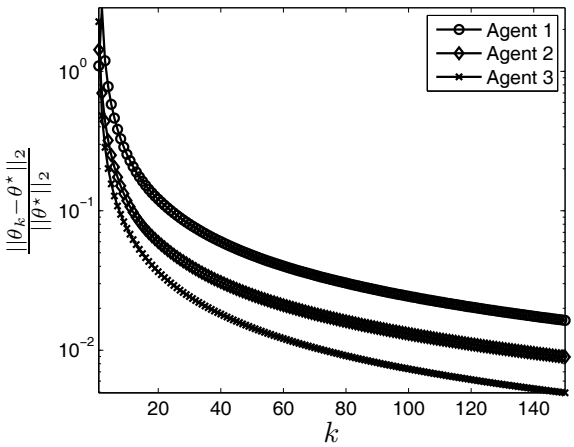


Figure 5.1: The consensus point estimates for each agent. The estimates are converging to $\theta^\star$, an optimizer of problem (5.4).

after a while. This behavior is typical for algorithms with diminishing stepsizes. For comparison, we solve problem (5.4) with θ fixed to the mean of the initial positions of the three agents, $\bar{\theta}$. The optimal cost is $\sum_{i=1}^{n} \nu_i(\theta^\star) = 6446$ and the cost for meeting at $\bar{\theta}$ is 6982. The corresponding optimal trajectories and control signals for agent 2 are shown in Figure 5.2 and Figure 5.3, respectively.

5.2 Faster Linear Iterations for Distributed Averaging

In the remainder of this chapter, we discuss distributed averaging (see also Section 2.3), which is a special class of distributed consensus. Distributed averaging has found diverse applications in areas such as multi-robot coordination, distributed synchronization and resource allocation in communication networks. In addition, as we have seen in Chapter 3, consensus iterations can be quite useful in distributed optimization.

Since the distributed consensus problem has received a large attention from several diverse research communities (from physics, mathematical programming to control and communications), it is natural that several different algorithms have been proposed. Many of these are linear iterations where each agent maintains one internal consensus state. Other algorithms are nonlinear, or have a larger state dimension than the basic iteration. One example of this is when the consensus is used for distributed quadratic programming where nodes maintain states corresponding to primal and dual (Lagrange multiplier) iterates. The iteration is then still linear, but of higher state dimension than the basic distributed averaging algorithm.

The objective of this section is to try to understand the potential benefits of

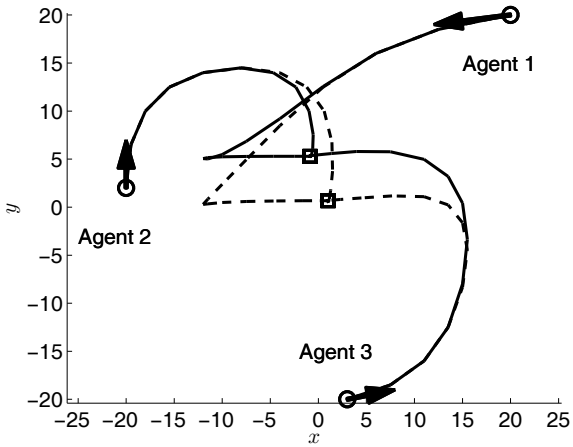


Figure 5.2: The trajectories of three agents with double integrator dynamics. The solid lines correspond to the optimal case, $\theta^\star$, and the dashed lines correspond to the mean case, $\bar{\theta}$. The circles are the starting points, the squares are the ending points, and the arrows show the initial velocities.

linear averaging iterations with extended state dimension. We establish conditions for convergence and explore how the parameters can be optimized to obtain faster convergence.

We consider systems of the following type

$$\begin{cases} x^{(k+1)} = Ax^{(k)} + Bu \\ y^{(k+1)} = Cx^{(k+1)}, \end{cases} \tag{5.8}$$

where $x^{(k)}$ is a vector of the nodes' internal states; $y^{(k)}$ is a vector of the nodes' outputs; and u is a vector with constant input. Each node i has an initial value, $[z]_i$, and the asymptotic output for each node should be the mean value of all nodes' starting values, $\lim_{k\to\infty}[y]_i(k) = \sum_{j=1}^{n}[z]_j/n$. Furthermore, the matrices A, B, and C respect the communication topology of the network, so that the nodes need to rely on only peer-to-peer communication to execute the iteration, as defined in Section 2.3. As always in this thesis, the network is composed of n nodes and is assumed to be connected.

In Section 2.3, we noted that it is possible to use different notions of convergence rates to quantify how fast the different algorithms approaches the desired output or fixed point. In this section, we focus on the geometric average convergence factor,

$$\lim_{k\to\infty}\left(\sup_{x\neq 0}\frac{||A^k x||_2}{||x||_2}\right)^{1/k} = \rho(A).$$

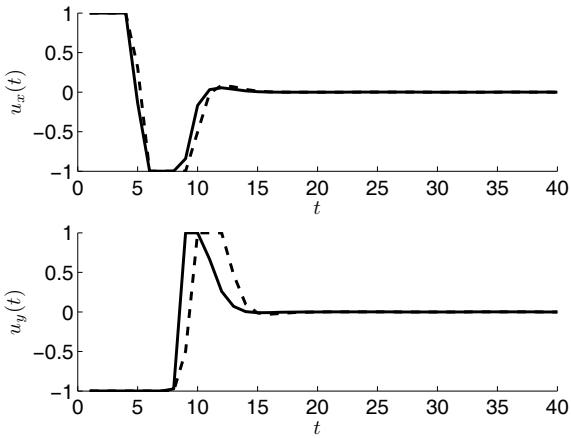


Figure 5.3: The control signals for agent 2, where $u_x(t)$ and $u_y(t)$ correspond to the acceleration in the x and y directions, respectively. The solid lines correspond to the optimal case and the dashed lines correspond to the mean case.

5.2.1 Background

As mentioned in Section 2.3, the consensus problem was pioneered by DeGroot (1974) and a seminal framework was introduced by Tsitsiklis (1984). In addition, there is a long tradition in Numerical Analysis to devise linear iterations (the consensus algorithm is a special case of this general class) to solve systems of linear equations; see, e.g., Varga (1962) for an early summary. It is also known that higher-order methods can accelerate many such linear methods (Young, 1972). Furthermore, consensus iterations can also be used for load balancing multiprocessor systems (Cybenko, 1989). In the load balancing literature, non-negative doubly stochastic matrices are most often used, which is restrictive but opens up for simple analysis using the powerful Perron-Frobenius theory.

Xiao and Boyd (2004) consider the linear iteration

$$x^{(k+1)} = Ax^{(k)} \tag{5.9}$$

and provide necessary and sufficient conditions on the matrix A for the iteration to converge to the average of the initial values. The average is reached if A fulfills

$$\lim_{k\to\infty} A^k = \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\intercal.$$

For the case of a symmetric A, they also provide a convex optimization problem formulation to find an A that minimizes the spectral radius. However, it is possible to do better using a slightly modified algorithm: if we allow scaling of the input, then it is possible to get the spectral radius arbitrarily close to zero (Olshevsky and Tsitsiklis, 2006). They let $\lim_{k\to\infty} A^k = \mathbf{1}_n\pi^\intercal$, where π is the stationary distribution of an

associated Markov chain, and use the initial state $[x_{\text{new}}^{(0)}]_i = [x^{(0)}]_i/([\pi]_i n)$. Hence, we have that $\lim_{k\to\infty} A^k x_{\text{new}}^{(0)} = \mathbf{1}_n \sum_{i=1}^n [x^{(0)}]_i [\pi]_i/([\pi]_i n) = \mathbf{1}_n \sum_{i=1}^n [x^{(0)}]_i/n$, which is the desired result. However, when the spectral radius goes to zero, so does π and we will have numerical difficulties. Olshevsky and Tsitsiklis (2006) also provide simple and numerically stable algorithms that have good worst case performance.

It is also possible to reach similar iterations by a completely different approach. Namely, if we view the average value as the optimal solution to a least-squares problem,

$$\begin{array}{ll} \underset{\{x_i\}_{i=1}^n}{\text{minimize}} & \sum_{i=1}^n \frac{1}{2}(x_i - x_i(0))^2 \\ & x_i = x_j,\ \forall (i,j) \in \mathcal{E}, \end{array} \tag{5.10}$$

then a distributed optimization algorithm solving this problem will also be a distributed averaging algorithm. Several optimization algorithms can be used, e.g., dual relaxation combined with a subgradient method (Rabbat et al., 2005) or the alternating direction multiplier method (Schizas et al., 2007). With a carefully posed optimization problem, both of the methods result in iterations of the same type as in (5.8). In addition, these alternative formulations can be used to devise iterations with bolstered resilience to communication noise, as shown in Schizas et al. (2007). This extension can be useful, since the iteration (5.9) is sensitive to communication noise due to its eigenvalue at 1 for A. However, the algorithm presented in Schizas et al. (2007) has the problem that internal values in the nodes grow without bound, since the desired fixed point is encoded in $x^{(k+1)} - x^{(k)}$. Finally, it is also possible to use a completely different approach to reach consensus. For example, Barbarossa and Scutari (2007) obtain a maximum likelihood estimator through local nonlinear coupling of first order dynamical systems.

5.2.2 General Shift-Register Case

Shift-registers can be used to speed up convergence in the stochastic version of (5.9) (Cao et al., 2006), and shift-registers are known to speed up several types of other deterministic iterative methods as well (Young, 1972).

For the consensus iteration case, shift-registers result in iterations of the type

$$\begin{cases} x^{(k+1)} = \begin{pmatrix} \beta A_{11} & (1-\beta) I_n \\ I_n & 0 \end{pmatrix} x^{(k)},\ x(0) = \begin{pmatrix} I_n \\ I_n \end{pmatrix} z \\ y^{(k+1)} = \begin{pmatrix} I_n & 0 \end{pmatrix} x^{(k+1)}, \end{cases} \tag{5.11}$$

where β is a constant scalar, A_{11} is a constant matrix (a matrix coming from (5.9) can be used). Furthermore, the matrix A_{11} should respect the communication topology, and therefore, we require that $A_{11} \in \mathcal{W}$, where $\mathcal{W}$ is defined in (2.8) on

page 18. The limit (if it exists) has the structure

$$\lim_{k\to\infty}\begin{pmatrix}\beta A_{11} & (1-\beta)I_n\\ I_n & 0\end{pmatrix}^k = \begin{pmatrix}\alpha\Delta & (1-\alpha)\Delta\\ \alpha\Delta & (1-\alpha)\Delta\end{pmatrix},$$

where α is a function of β. If A_{11} is symmetric, then it is possible, as Gosh et al. (1996) point out, to use a classical result by Golub and Varga (1961) to compute the optimal β.

Lemma 5.2.1 (Gosh et al. (1996) and Golub and Varga (1961))**.** *If $A_{11} = A_{11}^\intercal$, $A_{11}\mathbf{1}_n = \mathbf{1}_n$, and $\rho(A_{11} - \mathbf{1}_n\mathbf{1}_n^\intercal) = \mu < 1$, then the geometric average convergence factor for* (5.11) *is minimized if*

$$\beta^\star = \frac{2}{1+\sqrt{1-\mu^2}}.$$

The optimal geometric average convergence factor is

$$\rho\left(\begin{pmatrix}\beta^\star A_{11} & (1-\beta^\star)I_n\\ I_n & 0\end{pmatrix}\right) = \sqrt{\frac{1-\sqrt{1-\mu^2}}{1+\sqrt{1-\mu^2}}} < \mu, \quad 0<\mu<1.$$

Note that with $\beta^\star$, (5.11) will always asymptotically perform better than (5.9).

It is also possible to find the optimal β for a nonsymmetric A_{11}, but this is more complicated. The details of this method can be found in Manteuffel (1977) and Manteuffel (1982), and the two main steps are the following: first, the eigenvalues of A_{11} has to be enclosed by an ellipse with special properties. Second, using the parameters for this ellipse, the optimal β can be found.

The generalized version of (5.11), with m copies of the initial state, is the following

$$\begin{cases} x^{(k+1)} = Ax^{(k)},\ x(0) = \mathbf{1}_m \otimes z\\ y^{(k+1)} = \begin{pmatrix} I_n & \mathbf{1}_{m-1}^\intercal \otimes 0\end{pmatrix} x^{(k+1)},\end{cases} \tag{5.12}$$

where $A \in \mathbb{R}^{mn\times mn}$. To describe this generalized version in terms of (5.8), we have that $B = \mathbf{1}_m \otimes 0$ and $C = \begin{pmatrix} I_n & \mathbf{1}_{m-1}^\intercal \otimes 0_n\end{pmatrix}$. Furthermore, in order for the asymptotic output to reach the desired average, A needs to satisfy the limit

$$\lim_{k\to\infty} A^k = \frac{1}{n}\begin{pmatrix}\mathbf{1}_n\\ \vdots\\ \mathbf{1}_n\end{pmatrix}\begin{pmatrix}\alpha_1\mathbf{1}_n^\intercal & \dots & \alpha_m\mathbf{1}_n^\intercal\end{pmatrix}, \ \sum_{i=1}^m \alpha_i = 1, \tag{5.13}$$

since then

$$\lim_{k\to\infty} y^{(k)} = \frac{1}{n}\sum_{i=1}^m \alpha_i \mathbf{1}_n\mathbf{1}_n^\intercal z = \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\intercal z.$$

We have the following theorem (similar to Theorem 2 in Xiao and Boyd (2004)).

Theorem 5.2.2. *The iteration* (5.12) *satisfies* (5.13) *if and only if* A *and* α *fulfill the following conditions.*

a)

$$Af = f,\ f = \frac{1}{n}\begin{pmatrix}\mathbf{1}_n^\intercal & \dots & \mathbf{1}_n^\intercal\end{pmatrix}^\intercal. \tag{5.14}$$

b)

$$g^\intercal(\alpha)A = g^\intercal(\alpha),\ g^\intercal(\alpha) = \begin{pmatrix}\alpha_1\mathbf{1}_n^\intercal & \dots & \alpha_m\mathbf{1}_n^\intercal\end{pmatrix},\ \sum_{i=1}^{m}\alpha_i = 1. \tag{5.15}$$

c)

$$\rho\left(A - fg^\intercal(\alpha)\right) < 1, \tag{5.16}$$

where $\rho(\cdot)$ *denotes the spectral radius.*

Proof. We start with showing sufficiency of the conditions. If conditions a) and b) are satisfied, then we have that

$$\begin{aligned}(A - fg^\intercal(\alpha))^k &= (A - Afg^\intercal(\alpha))^k = A^k(I - fg^\intercal(\alpha))^k \\ &= A^k(I - fg^\intercal(\alpha)) = A^k - fg^\intercal(\alpha),\end{aligned}$$

where we used $Af = f$ in the first equality, and the third equality is based on the fact that $(I - fg^\intercal(\alpha))(I - fg^\intercal(\alpha)) = I - 2fg^\intercal(\alpha) + fg^\intercal(\alpha)fg^\intercal(\alpha) = I - 2fg^\intercal(\alpha) + fg^\intercal(\alpha)(1/n)\sum_{i=1}^{m} n\alpha_i = I - fg^\intercal(\alpha)$. Now condition c) implies that $\lim_{k\to\infty} A^k - fg^\intercal(\alpha) = \lim_{k\to\infty}(A - fg^\intercal(\alpha))^k = 0$, and sufficiency is established. We continue with necessity. The limit $\lim_{k\to\infty} A^k = fg^\intercal(\alpha)$ exists if and only if there exist an invertible matrix T, a matrix Z with $\rho(Z) < 1$, and $\kappa \geq 0$, such that

$$A = T\begin{pmatrix} I_\kappa & 0 \\ 0 & Z\end{pmatrix}T^{-1}, \tag{5.17}$$

where I_κ is the κ-dimensional identity matrix (Meyer and Plemmons, 1977). When the limit exists, we have

$$\lim_{k\to\infty} A^k = T\begin{pmatrix} I_\kappa & 0 \\ 0 & 0\end{pmatrix}T^{-1} = fg^\intercal(\alpha),$$

and since $\operatorname{rank} fg^\intercal(\alpha) = 1$, we know that $\operatorname{rank} I_\kappa = 1$ and $\kappa = 1$. The limit and (5.17) also imply that $fg^\intercal(\alpha)A = Afg^\intercal(\alpha) = fg^\intercal(\alpha)$, thus f is a right eigenvector and g is a left eigenvector, both with eigenvalue 1. Finally, we also have that

$$\rho(A - fg^\intercal(\alpha)) = \rho\left(T\begin{pmatrix} 0 & 0 \\ 0 & Z\end{pmatrix}T^{-1}\right) < 1.$$

We conclude that the conditions are both necessary and sufficient. □

5.2.3 Minimizing the Spectral Radius

As mentioned previously, the spectral radius is crucial for the convergence rate. Thus, it is of interest to find an A with minimal spectral radius. Optimization of the spectral radius is very hard in general, but in the case of a symmetric (Hermitian in the most general case) A, the spectral norm and the spectral radius coincide; the spectral norm is convex and therefore much easier to work with. The symmetric case is discussed in Section 2.3.

One potential way to find a minimizing nonsymmetric matrix is to use scalings; we let $A = T\widehat{A}T^{-1}$ with $\widehat{A} = \widehat{A}^{\mathsf{T}}$, and the optimization problem will, under some conditions on T, boil down to a symmetric optimization problem over $\widehat{A}$. However, it is not known which scalings, T, should be used and if the optimal A can be written as $A = T\widehat{A}T^{-1}$. This approach is therefore not so useful. Instead, we can directly look for a nonsymmetric matrix (for a fixed α in condition b) in Theorem 5.2.2) that satisfies Theorem 5.2.2 and the sparsity constraints.

For our developments, define the error, $\tilde{x}(k)$, as $\tilde{x}^{(k)} = x^{(k)} - x^\star$, where $x^\star$ is the fixed point of the linear iteration (5.12). We have the following iteration for $\tilde{x}^{(k)}$,

$$\tilde{x}^{(k+1)} = x^{(k+1)} - x^\star = Ax^{(k)} - x^\star = A(x^{(k)} - x^\star) = A\tilde{x}^{(k)} = (A - fg^{\mathsf{T}}(\alpha))\tilde{x}^{(k)}.$$

The output should converge to the correct asymptotic output as fast as possible, and the asymptotic convergence factor, $\lim_{k\to\infty}\big(\sup_{\|x^{(0)}\|=1}\big\|C\tilde{x}^{(k)}\big\|\big)^{(1/k)}$, fulfills the following

$$\lim_{k\to\infty}\left(\sup_{\|x^{(0)}\|=1}\left\|C\tilde{x}^{(k)}\right\|\right)^{(1/k)} = \lim_{k\to\infty}\left\|C(A - f(g\alpha))^k\right\|^{(1/k)} \leq \rho\big(A - fg(\alpha)\big).$$

Thus, if the output error should decay as fast as possible, we can achieve this by minimizing the spectral radius of $(A - fg^{\mathsf{T}}(\alpha))$, which, as previously mentioned, unfortunately is a very hard problem. We will, however, try to develop a useful procedure. To this end, we need the following lemma (see, e.g., Horn and Johnson (1985, Problem 25, Section 5.6)).

Lemma 5.2.3. *If $X \in \mathbb{R}^{n\times n}$ and $\varepsilon > 0$, then there exists a matrix valued function $S(X,\varepsilon) \in \mathbb{R}^{n\times n}$ such that*

$$\rho(X) \leq \big\|S(X,\varepsilon)^{-1}XS(X,\varepsilon)\big\| \leq \rho(X) + \varepsilon,$$

where

$$\big\|S(X,\varepsilon)^{-1}XS(X,\varepsilon)\big\| = \max_{\|S(X,\varepsilon)^{-1}u\|=1}\big\|S(X,\varepsilon)^{-1}Xu\big\| \triangleq \|X\|_{S(X,\varepsilon)^{-1}}.$$

Since $\rho(x) \leq |||X|||$ for all matrix norms $|||\cdot|||$, we have that $\inf\{\|X\|_S \,|S > 0\} = \rho(X)$. Thus, if we minimize the norm $\|A - fg^{\mathsf{T}}(\alpha)\|_S$, and let S be a decision variable as well, we can get arbitrarily close to the spectral radius. The following lemma (see, e.g, Horn and Johnson (1985, Theorem 7.7.6)) will be useful.

Lemma 5.2.4 (Schur complement)**.** *For $U, V, X \in \mathbb{R}^{n\times n}$, the following holds*

$$\begin{cases} U > 0 \\ X > V^\intercal U^{-1} V \end{cases} \Leftrightarrow \begin{pmatrix} U & V \\ V^\intercal & X \end{pmatrix} > 0.$$

Note that $\|A - fg^\intercal(\alpha)\|_S \leq \psi$ is equivalent with

$$(A - fg^\intercal(\alpha))^\intercal Q(A - fg^\intercal(\alpha)) \leq \psi Q$$

with S nonsingular and $Q = S^\intercal S$. From the above discussion and lemmas, we have that

$$\begin{aligned} \underset{A,\psi,S}{\text{minimize}} \quad & \psi \\ \text{subject to} \quad & \|A - fg^\intercal(\alpha)\|_S \leq \psi \\ & S \text{ nonsingular} \\ & Af = f, \quad A \in \mathcal{W} \\ & Ag^\intercal(\alpha) = g^\intercal(\alpha) \end{aligned}$$

is equivalent with

$$\underset{A,\psi,Q,\Upsilon}{\text{minimize}} \quad \psi \tag{5.18a}$$

$$\text{subject to} \begin{pmatrix} \psi Q & Q(A-\Upsilon) \\ (A-\Upsilon)^\intercal Q & Q \end{pmatrix} \geq \begin{pmatrix} \delta I & 0 \\ 0 & \delta' I \end{pmatrix} \tag{5.18b}$$

$$\Upsilon = A - fg^\intercal(\alpha), \quad Q^\intercal = Q, \quad Q \geq \delta' I \tag{5.18c}$$

$$Af = f, \quad A \in \mathcal{W} \tag{5.18d}$$

$$Ag^\intercal(\alpha) = g^\intercal(\alpha), \tag{5.18e}$$

for some $\delta' > 0$ and $\delta > 0$. The constraint (5.18b) is a so-called bilinear matrix inequality (BMI), which is a nonconvex constraint. If we can solve this type of problems, then we can find a nonsymmetric A that minimizes $||A - fg^\intercal(\alpha)||_S$, and thereby also pushing down the spectral radius of $(A - fg^\intercal(\alpha))$, while still satisfying Theorem 5.2.2 and the sparsity constraints. It is hard to find a globally optimal solution, but some software packages can generate local solutions, as we will see in Section 5.2.4.

5.2.4 Numerical Examples

To evaluate the performance of the different linear iterations that achieve distributed averaging, we consider the simple network in Fig. 5.4. To avoid numerical difficulties and maintain computational tractability, the network consists of only 7 nodes. We investigate four methods of finding the system matrices, using the parser YALMIP (Löfberg, 2004), and the solvers SDPT3 (Tutuncu et al., 2003) and PENBMI (Kocvara and Stingl, 2003).

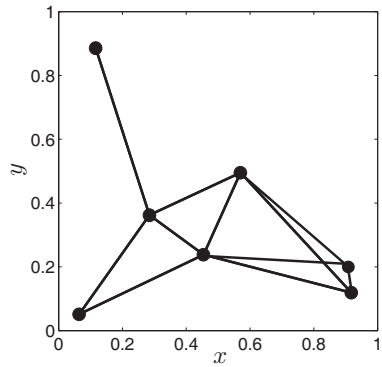


Figure 5.4: Topology of the random 7-node network.

Symmetric Matrix Algorithm. Dimension $n \times n$.

We use the method described in Section 2.3 in conjunction with YALMIP and SDPT3 to find the symmetric A that minimizes the spectral norm (which is the same as the spectral radius in this case).

Nonsymmetric Matrix Algorithm. Dimension $n \times n$.

The method outlined in Section 5.2.3 is used in conjunction with YALMIP and PENBMI to find a nonsymmetric matrix that is a local minimum of the matrix norm $||A - \mathbf{1}_n \mathbf{1}_n^\intercal / n||_S$, and since we are optimizing over S as well, this norm is close to the spectral radius, as previously discussed.

Shift-Register Algorithm. Dimension $2n \times 2n$.

Consider the shift-register example (with a fixed matrix A_{11} and m copies of the state) once again. The structure of A is

$$A = \begin{pmatrix} \beta_1 A_{11} & \dots & \beta_{m-1} I & \beta_m I \\ I & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & I & 0 \end{pmatrix}.$$

With this structure and $\sum_{i=1}^{m} \beta_i = 1$, condition a) in Theorem 5.2.2 is satisfied. To satisfy condition b) in Theorem 5.2.2, α and β need to satisfy,

$$\begin{aligned}
\alpha_m &= \alpha_1 \beta_m, \\
\alpha_{m-1} &= \alpha_1 \beta_{m-1} + \alpha_m, \\
\alpha_{m-2} &= \alpha_1 \beta_{m-2} + \alpha_{m-1}, \\
&\vdots \\
\alpha_1 &= \alpha_1 \beta_1 + \alpha_2,
\end{aligned}$$

in addition with $\sum_{i=1}^{m} \alpha_i = 1$. Finally, to satisfy condition c) in Theorem 5.2.2, α, β, and A need to satisfy

$$\rho\left(A(\beta) - fg^{\mathsf{T}}(\alpha)\right) < 1.$$

We use the above conditions with $m = 2$ (this means we have one shift-register and one scalar weight to choose, namely β) and use the optimal symmetric A matrix devised using the method in Section 2.3 as our A_{11} matrix. The optimal β is given by Lemma 5.2.1, and we have that $\beta^\star = 1.1716$.

General Shift-Register Algorithm. Dimension $2n \times 2n$.

Here we use the f and g vectors from the previous section ($m = 2$ and $\beta = 1.1716$) and search for a general nonsymmetric A using the method in Section 5.2.3, with

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad A_{11}, A_{12}, A_{21}, A_{22} \in \mathcal{W}.$$

It is possible to do a search over β, but it is not really tractable, since the optimization with BMI is rather time consuming.

Results

For the communication topology shown in Fig. 5.4, the performance of the four resulting algorithms are shown in Fig. 5.5. We use the following performance metrics: for the $n \times n$ matrices

$$\text{performance}_{n\times n}(k) = \left\| A^k - \mathbf{1}_n 1_n^{\mathsf{T}}/n \right\|_2, \tag{5.19}$$

which is the worst case norm of the deviation from the desired vector. For the $2n \times 2n$ matrices, we have

$$\text{performance}_{2n\times 2n}(k) = \left\| \begin{pmatrix} I_n & 0 \end{pmatrix} A^k \begin{pmatrix} I_n \\ I_n \end{pmatrix} - \mathbf{1}_n 1_n^{\mathsf{T}}/n \right\|_2, \tag{5.20}$$

where we only consider the worst case for the first n elements in the vector, since this part of the vector constitute the output ($C = \begin{pmatrix} I_n & 0 \end{pmatrix}$).

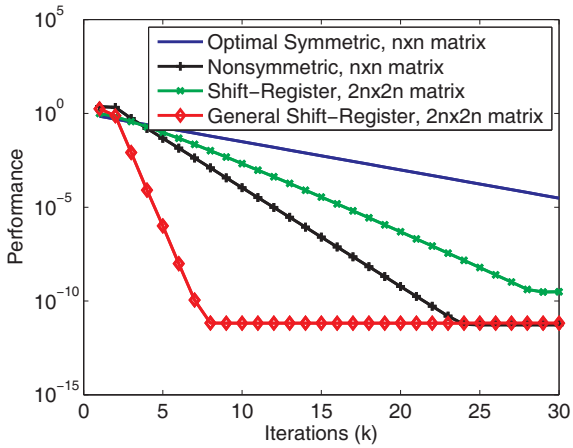


Figure 5.5: Performance of the four algorithms for a 7 node network.

The general shift-register algorithm has significantly better performance than the other algorithms. The nonsymmetric matrix algorithm is second best, and the shift-register algorithms is quite close. The symmetric matrix algorithm has the worst performance in this example. All four algorithms flattens out, which is not predicted by the theory. We believe that complete convergence is hindered by numerical problems.

Thus, this example suggests that much can be gained from using the general shift-register algorithm. However, since this optimization problem is bilinear, it is computationally intractable for topologies with more than ten nodes. The same is valid for the nonsymmetric matrix algorithm.

The most computationally viable way to increase the convergence speed is thus the shift-register algorithm from Section 5.2.4. In Fig. 5.6, we show Monte Carlo simulations for the shift-register algorithm and the optimal symmetric algorithm. For each number of nodes, 1000 connected random networks were generated, and the mean of the performance (using (5.19) and (5.20)) for $k = 5, 25, 125$ is shown. The simulations indicate that the shift-register algorithm is better if high accuracy is desired, while the optimal symmetric algorithm has better initial convergence speed.

Thus, a performance boost can be achieved by using the simple shift-register algorithm, with virtually no extra computational burden on the nodes nor on the off-line algorithm that computes the system matrices.

5.2.5 General Case

We now look at necessary and sufficient conditions for a more general case of (5.8) to converge to the average of the starting values, while respecting the communication

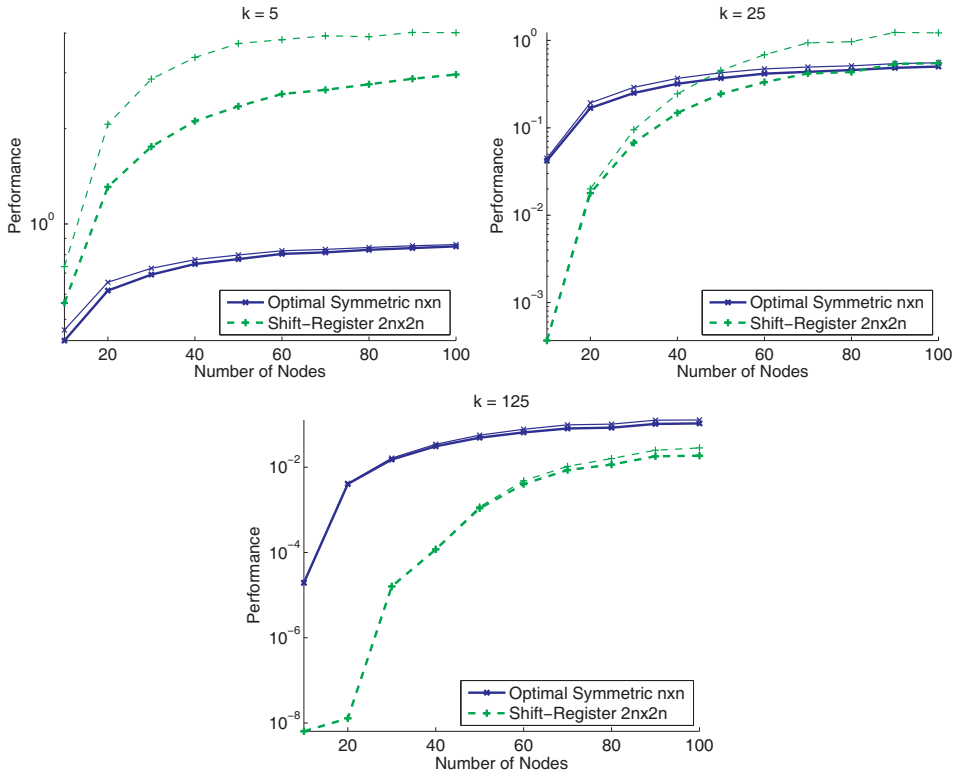


Figure 5.6: Average performance of the Symmetric Matrix Algorithm and the Shift-Register Algorithm. For each number of nodes, 1000 random networks were generated and averaged over. The thick line denotes the average and the thin line denotes the sum of the average and the computed variance.

constraints between the nodes.

We consider the following system

$$\begin{cases} x^{(k+1)} &= Ax^{(k)} + Bz \\ y^{(k+1)} &= Cx^{(k+1)} \\ x(0) &= Ez, \end{cases} \tag{5.21a}$$

with the state $x^{(k)} \in \mathbb{R}^{nm}$, the output $y^{(k)} \in \mathbb{R}^n$, the starting values $z \in \mathbb{R}^n$, and

$$A = \begin{pmatrix} A_{11} & \dots & A_{1m} \\ \vdots & \ddots & \vdots \\ A_{m1} & \dots & A_{mm} \end{pmatrix}, \; A_{ij} \in \mathcal{W}, \; \forall i, j = 1, ..., m, \tag{5.21b}$$

$$B = \begin{pmatrix} B_1^\mathsf{T} & \dots & B_m^\mathsf{T} \end{pmatrix}^\mathsf{T}, \, B_i \in \mathbb{R}^{n\times n} \text{ and diagonal,} \tag{5.21c}$$

$$C = \begin{pmatrix} C_1 & \dots & C_m \end{pmatrix}, \, C_i \in \mathbb{R}^{n\times n} \text{ and diagonal,} \tag{5.21d}$$

$$E = \begin{pmatrix} E_1^\mathsf{T} & \vdots & E_m^\mathsf{T} \end{pmatrix}^\mathsf{T}, \, E_i \in \mathbb{R}^{n\times n} \text{ and diagonal.} \tag{5.21e}$$

We need the following definition (see, e.g., Ben-Israel and Greville (1974)) to get an elegant description of the limit of (5.21).

Definition 5.2.5. *Given a matrix $A \in \mathbb{R}^{n\times n}$, the Drazin pseudoinverse is the matrix $A^\mathrm{d} \in \mathbb{R}^{n\times n}$ fulfilling*

$$AA^\mathrm{d} = A^\mathrm{d}A, \quad A^{k+1}A^\mathrm{d} = A^k, \quad A(A^\mathrm{d})^2 = A^\mathrm{d},$$

for some integer k in the interval $0 \leq k \leq n$.

The Drazin pseudoinverse A^d always exists (for symmetric A) and it is unique. We have the following lemma (Ben-Israel and Greville, 1974, Theorem 4, page 164).

Lemma 5.2.6. *If $A \in \mathbb{R}^{n\times n}$ fulfills $\operatorname{rank} A = \operatorname{rank} A^2$ and $A = TJT^{-1}$ where J is a Jordan normal form of A, then*

$$A^\mathrm{d} = TJ^\dagger T^{-1},$$

where $J^\dagger$ is the Moore-Penrose pseudoinverse of J.

Note that if A is invertible then $A^{-1} = A^\dagger = A^\mathrm{d}$. The following lemma will be helpful in the convergence analysis (cf. Meyer and Plemmons (1977)).

Lemma 5.2.7. *If $A \in \mathbb{R}^{nm\times nm}$ can be written on the Jordan normal form $A = TJT^{-1}$ with*

$$J = \begin{pmatrix} I_\kappa & 0 \\ 0 & Z \end{pmatrix}$$

with $\rho(Z) < 1$ and $\kappa \in \mathbb{N}$, $0 \leq \kappa \leq nm$. Then $x^{(k)}$ generated by (5.21) fulfills

$$x^{(k)} = A^k Ez + kT \begin{pmatrix} I_\kappa & 0 \\ 0 & 0 \end{pmatrix} T^{-1}Bz + T \begin{pmatrix} 0 & 0 \\ 0 & \sum_{j=0}^{k-1} Z^j \end{pmatrix} T^{-1}Bz, \tag{5.22}$$

where

$$\lim_{k\to\infty} T \begin{pmatrix} 0 & 0 \\ 0 & \sum_{j=0}^{k-1} Z^j \end{pmatrix} T^{-1} = T \begin{pmatrix} 0 & 0 \\ 0 & (I-Z)^{-1} \end{pmatrix} T^{-1} = (I-A)^\mathrm{d}$$

and

$$\lim_{k\to\infty} A^k = T \begin{pmatrix} I_\kappa & 0 \\ 0 & 0 \end{pmatrix} T^{-1} = I - (I-A)(I-A)^\mathrm{d}.$$

Proof. The equation (5.22) follows directly from (5.21) and the assumptions. Since $\rho(Z) < 1$, we have $\lim_{k\to\infty} \sum_{j=0}^{k} Z^j = (I_{nm-\kappa} - Z)^{-1}$ and

$$T \begin{pmatrix} 0 & 0 \\ 0 & (I-Z)^{-1} \end{pmatrix} T^{-1} = T \begin{pmatrix} 0 & 0 \\ 0 & (I-Z) \end{pmatrix}^{\dagger} T^{-1} = T(I-J)^{\dagger}T^{-1} = (I-A)^{\mathrm{d}},$$

where the last equality follows from Lemma 5.2.6. Furthermore, since $\rho(Z) < 1$, we have $\lim_{k\to\infty} Z^k = 0$ and

$$\begin{aligned} I - (I-A)(I-A)^{\mathrm{d}} &= T \left(I - \left(I - \begin{pmatrix} I & 0 \\ 0 & Z \end{pmatrix} \right) \begin{pmatrix} 0 & 0 \\ 0 & (I-Z)^{-1} \end{pmatrix} \right) T^{-1} \\ &= T \begin{pmatrix} I_\kappa & 0 \\ 0 & 0 \end{pmatrix} T^{-1}. \end{aligned}$$

□

We have the following theorem.

Theorem 5.2.8. *Consider the system defined by* (5.21)*. The limit* $\lim_{k\to\infty} A^k$ *exists, the sequence of states,* $\{x^{(k)}\}$*, converges, and the corresponding system output satisfies* $\lim_{k\to\infty} y^{(k)} = \mathbf{1}_n \mathbf{1}_n^{\intercal} z/n$ *for all* $z \in \mathbb{R}^n$ *if and only if the matrices* A*,* B*,* C*, and* E *satisfy the following conditions:*

a) There exist two matrices, $T \in \mathbb{R}^{nm\times nm}$ *and* Z*, and an integer,* κ*, such that*

$$A = T \begin{pmatrix} I_\kappa & 0 \\ 0 & Z \end{pmatrix} T^{-1},\ \rho(Z) < 1,\ 0 \le \kappa \le nm. \tag{5.23}$$

b)

$$\mathcal{R}(B) \subseteq \mathcal{N}(A-I)^{\perp}, \tag{5.24}$$

where $\mathcal{R}(B)$ *denotes the range of the matrix* B *and* $\mathcal{N}(A-I)^{\perp}$ *denotes the orthogonal complement to the nullspace of the matrix* $A - I$*.*

c)

$$C\left(\left(I - (I-A)(I-A)^{\mathrm{d}}\right)E + (I-A)^{\mathrm{d}}B\right) = \mathbf{1}_n\mathbf{1}_n^{\intercal}/n. \tag{5.25}$$

Proof. We start with the *if* part. The condition (5.23) implies that $\lim_{k\to\infty} A^k$ exists (Meyer and Plemmons, 1977).

From Lemma 5.2.7 we have that

$$x^{(k)} = A^k E z + kT \begin{pmatrix} I_\kappa & 0 \\ 0 & 0 \end{pmatrix} T^{-1} B z + T \begin{pmatrix} 0 & 0 \\ 0 & \sum_{j=0}^{k-1} Z^j \end{pmatrix} T^{-1} B z.$$

Note that $T\left(\begin{smallmatrix} I_\kappa & 0 \\ 0 & 0 \end{smallmatrix}\right)T^{-1}$ has the same eigenvectors as A, and we have that $Au = u$ implies $T\left(\begin{smallmatrix} I_\kappa & 0 \\ 0 & 0 \end{smallmatrix}\right)T^{-1}u = u$ and $Au = \lambda u$ with $|\lambda| < 1$ implies $T\left(\begin{smallmatrix} I_\kappa & 0 \\ 0 & 0 \end{smallmatrix}\right)T^{-1}u = 0$. Furthermore, condition (5.24) implies that $kT\left(\begin{smallmatrix} I_\kappa & 0 \\ 0 & 0 \end{smallmatrix}\right)T^{-1}Bz = 0$ for all z, since (5.24) implies that the columns in B are orthogonal to the κ eigenvectors with eigenvalue 1 of $T\left(\begin{smallmatrix} I_\kappa & 0 \\ 0 & 0 \end{smallmatrix}\right)T^{-1}$.

Using Lemma 5.2.7 again, we now have

$$\lim_{k\to\infty} x^{(k)} = \left(\left(I - (I-A)(I-A)^{\mathrm{d}}\right)E + (I-A)^{\mathrm{d}}B\right)z,$$

which in combination with condition (5.25) yields

$$\lim_{k\to\infty} y^{(k)} = C\left(\left(I - (I-A)(I-A)^{\mathrm{d}}\right)E + (I-A)^{\mathrm{d}}B\right)z = \mathbf{1}_n\mathbf{1}_n^{\intercal}/nz, \quad \forall z \in \mathbb{R}^n.$$

It is clear that $\{x^{(k)}\}$ converges and $\lim_{k\to\infty} y^{(k)} = \mathbf{1}_n\mathbf{1}_n^{\intercal}/nz$ for all $z \in \mathbb{R}^n$.

Let us continue with the *only if* part. From Meyer and Plemmons (1977) we know that existence of $\lim_{k\to\infty} A^k$ implies (5.23). Existence of $\lim_{k\to\infty} A^k$ and convergence of $\{x^{(k)}\}$ for all $z \in \mathbb{R}^n$ implies that $\|T\left(\begin{smallmatrix} I_\kappa & 0 \\ 0 & 0 \end{smallmatrix}\right)T^{-1}B\|$ has to be zero, which implies (5.24). Finally, the output has to fulfill $\lim_{k\to\infty} y^{(k)} = \mathbf{1}_n\mathbf{1}_n^{\intercal}z/n$ for all $z \in \mathbb{R}^n$, which in combination with the above discussion implies that

$$\lim_{k\to\infty} y^{(k)} = C\left(\left(I - (I-A)(I-A)^{\mathrm{d}}\right)E + (I-A)^{\mathrm{d}}B\right)z = \mathbf{1}_n\mathbf{1}_n^{\intercal}/nz, \quad \forall z \in \mathbb{R}^n,$$

which, in turn, implies (5.25). □

Remark 5.2.9. The conditions in Theorem 5.2.8 are difficult to use in an optimization problem. For example, the Drazin inverse is hard to compute numerically.

Remark 5.2.10. It is possible to find matrices fulfilling the conditions in Theorem 5.2.8. One choice is the matrices from the algorithms in Section 5.2.4. Another choice is the matrices from a dual relaxation solution algorithm to (5.10).

5.3 Summary

Primal decomposition and incremental subgradient methods provide an interesting framework to pose distributed consensus problems. It allows us to consider general linear models for the agents and easily handle convex input constraints and linear state constraints. The convergence is mathematically guaranteed in the simplest case when negotiation and motion phases are separated.

Future work includes the extension of the results to strategies with interleaved negotiation and motion phases and realistic models of the communication network.

Motivated by the many different algorithms that have been proposed for distributed averaging, we have investigated the convergence of a more general class of linear averaging iterations. First, we considered linear iterations that allow nodes to maintain several internal states. We provided necessary and sufficient conditions

for convergence, and discussed some possible optimization schemes to improve the convergence rate.

It turns out that with a single shift-register, we can use a classical result to compute the optimal weights, giving a computationally inexpensive way to find a weight matrix A that converges faster (in the asymptotic sense) than the original algorithm in Xiao and Boyd (2004). A shift-register algorithm is better if high accuracy is desired, while the optimal symmetric algorithm has better initial convergence speed. In addition, numerical examples indicate that performance can be vastly improved if we use a nonsymmetric matrix A. However, this optimization problem is computationally intractable for larger network topologies.

For a truly distributed system, it is probably not possible to run any optimization problem at all to find a good (or optimal) A matrix, especially since the complete network topology needs to be known. In this case, the only viable methods seem to be the heuristic methods presented in Section 2.3, which require only local topology information. However, if the consensus iteration should be executed several times over the same graph, then it can be worthwhile to optimize the weighting matrix W.

Finally, we looked at necessary and sufficient conditions for the convergence to consensus of a larger class of linear iterations with an augmented state vector. These conditions are not amenable for optimization in their present form, but we are looking into this in our future work.

Chapter 6

Optimization in Wireless Sensor Networks

> "*While a serial algorithm consists of a single such system, a distributed algorithm consists of an interconnection of many such systems, yielding a composite system that can be quite complex to model, analyze, and implement.*"
>
> D. P. Bertsekas and J. N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods, 1997.

In this chapter, we detail how some of the algorithms previously presented can be implemented in a real networked system. More specifically, we would like to devise a generic optimization component to be used in WSNs. First, we start with simulations of the optimization algorithms in the network simulator NS2 (NS2, 2008). Then, we implement the algorithms on Tmote Sky motes (Tmote, 2008).

The optimization algorithms previously presented are rather mathematically clean and simple, at least in the author's opinion; still, as we will see, they require a substantial amount of work to be implemented in practice.

The chapter is organized as follows. We start with some general background in Section 6.1. In Section 6.2, we present the optimization problem, which should be solved in a WSN. It turns out that this optimization problem can be solved in several ways that give rise to different communication patterns, which is discussed in Section 6.3. Section 6.4 presents, in detail, the algorithms that will be evaluated. In Section 6.5, we describe how these algorithms were implemented in NS2. Then, in Section 6.6, we present how the optimization algorithms were implemented on Tmote Sky motes and we also report our experimental results from this implementation. Finally, we conclude the chapter with summary and discussion in Section 6.7.

6.1 Background

Several important tasks of WSNs, including estimation, detection, localization, and resource sharing, can be cast as optimization problems (Boyd and Vandenberghe,

2004). Recently, a large research effort has been devoted to understanding distributed quadratic programming problems and the properties of the associated distributed averaging or consensus algorithms, which we have previously discussed in Section 5.2. However, moving beyond the least-squares framework offers distinctive advantages in applications (see, e.g., Rabbat and Nowak (2004), Wang (2007), and Johansson et al. (2008*e*)) and the development of a generic optimization component for WSNs could potentially have a very broad applicability.

Bringing state-of-the art optimization solvers to the resource-constrained sensor networking environment is a grand challenge. Modern high-performance solvers require significant memory and computations and rely on central access to all problem data. In contrast, for sensor network applications one would like to find distributed algorithms which remove the single point of failure of having a central computational node. The resource-constrained environment demands algorithms with small computational and memory requirements. It is desirable that the traffic for coordinating nodes is minimal and that the communication can be done in a way that makes economic use of the wireless medium to reduce energy consumption. Finally, several emerging applications require that nodes do not reveal "private" information when coordinating their decisions with the others.

The idea of optimization in WSNs is of course not new and there are many suggestions on algorithms, such as Rabbat et al. (2005), Wang (2007), and Blatt et al. (2007). Rabbat et al. (2005) suggest a dual approach that we will come back to later in this chapter. On the other hand, Blatt et al. (2007) pertain to differentiable objective functions, which does not fit our purposes since we consider potentially non-differentiable objective functions, and Wang (2007) focuses on the fundamentally different approach of particle swarm optimization.

We will continue to focus on algorithms that are of peer-to-peer nature and do not rely on any networking functionality apart from nearest neighbor communications, as defined in Section 2.3 on page 17. Furthermore, the algorithms suggest rather different ways of coordinating nodes: from passing around a "best estimate" using unicast transmissions to having nodes asynchronously broadcasting their suggestion for the global variables. We implement a selection of promising algorithms in the network simulator NS2 (NS2, 2008) and perform extensive and detailed packet-level simulations of algorithm performance in wireless sensor networks equipped with 802.15.4 radios (see IEEE (2006)), studying convergence rate and algorithm accuracy. In addition, we implement the same algorithms on the Tmote Sky platform. Finally, we perform experiments on these Tmote Sky motes to assess the performance of the chosen algorithms.

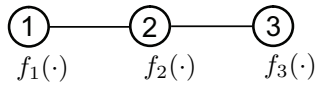


Figure 6.1: Example setup with three nodes. A line between nodes implies that they are connected.

6.2 Problem Formulation

We consider a convex optimization problem of the same type as in Chapter 3, i.e.,

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \sum_{i=1}^{n} f_i(x) \\ \text{subject to} & x \in \mathcal{X} \end{array} \tag{6.1}$$

where $f_i : \mathbb{R}^{\xi} \to \mathbb{R}$ is a convex cost function associated with node i and x is a global decision variable that all nodes should agree upon. Note that we can allow the node to have "private" optimization variables by letting $f_i(x) \triangleq \min_{y_i \in \mathcal{Y}_i} h_i(x, y_i)$ for some function h_i and set $\mathcal{Y}_i$, as long as f_i is convex. The feasible set $\mathcal{X}$ is assumed to be nonempty, convex, and closed. Associated to the problem is a communication topology represented by a graph $\mathcal{G}$ with edges $\mathcal{E}$ and vertices $\mathcal{V}$. The presence of an edge (i, j) in $\mathcal{G}$ means that node i can communicate (i.e., exchange information) directly with node j, and an example setup is illustrated in Fig. 6.1. The problem is to choose an optimization algorithm that can be easily implemented in a WSN, does not waste energy in the nodes, and does not introduce a lot of communication overhead.

6.3 A Taxonomy of Solution Approaches

There are many ways of structuring the computations for finding the optimal solution $x^{\star}$.

The most obvious is a *centralized* approach where nodes pass information about f_i (if private variables are used, then it may not be possible to find a closed form expression for f_i and the nodes have to pass h_i and $\mathcal{Y}_i$) to a central node that solves the convex optimization problem (6.1) using, e.g., the techniques in Boyd and Vandenberghe (2004), and then distributes the solution to the nodes. This solution is sensitive to a single point of failure (the central node), requires individual nodes to reveal their cost functions and constraint sets, and demands potentially complex information transfer from nodes to the computational unit.

It is often more attractive to use a *decentralized* approach, where nodes collaborate to find the optimal solution to the decision problem. One way to develop such approaches is to rely on decomposition techniques from mathematical programming, as we have done in Section 2 and in Section 4. Many decomposition methods ex-

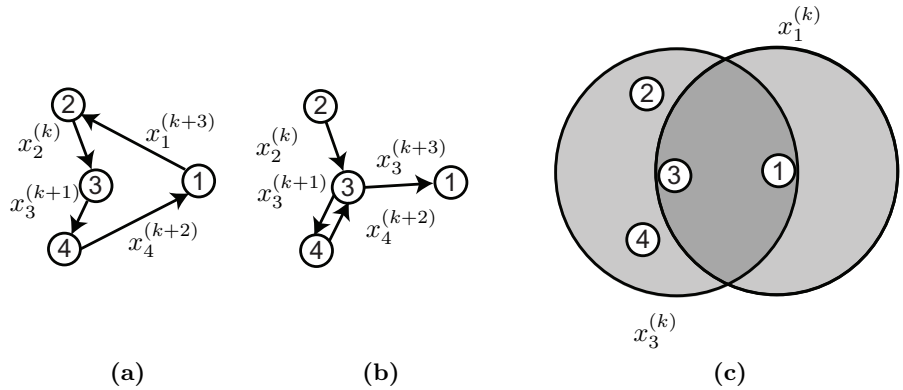


Figure 6.2: (a) The ring setup, where information is passed around in a logical ring. (b) The random neighbor setup, where each node unicast information to a random neighbor. (c) The broadcast setup, where each node broadcast information to their neighbors. The broadcast setup and the random neighbor setup are both peer-to-peer.

ist and they suggest very different ways of structuring the computations and the communication required for solving the decision problem.

One class of methods, based on incremental subgradient methods, which we have discussed in Section 3.3, result in *estimate passing* solutions: the nodes are organized in a ring, and the current estimate of the optimal solution is passed from one node to the other; see Fig. 6.2a. When a node receives the current estimate, it performs a local update accounting for its own cost functions and constraint, before passing the modified estimate to the next node. In this approach, nodes do not need to reveal their private objectives and constraints but the method requires reliable communications as the messages conveying the current estimate must not be lost. Other approaches suggest computations based on *peer-to-peer* communication and coordination where nodes exchange information only with neighbors; see Fig. 6.2b and Fig. 6.2c. No private information needs to be exchanged, and no network-wide signalling needs to be implemented. In Fig. 6.2b, the current estimate is circulated in the network; when a mote receives the estimate, it updates the estimate according to its local cost function and passes the estimate to a random neighbor. In Fig. 6.2c, all motes broadcast their current estimate to their neighbors at each iteration.

For the specific case when the nodes have no internal variables and the cost functions are quadratic,

$$f_i(x) = \frac{1}{2}(x - c_i)^2,\ i = 1, ..., n,$$

many special algorithms exist, since the optimal solution is the network-wide average of the constants c_i, which we discussed in detail in Section 5.2. Although the theory for the quadratic case, i.e., consensus algorithms, is rather well-developed, we

would like to stress that those algorithms do not, in general, apply to our problem formulation (6.1).

6.3.1 The Impact of Wireless Communications

In our classification above, communication has been modeled in a very crude and abstract way. However, for wireless devices, the precise details of the communication strategy are critical and can have a large influence on message latency and node energy consumption. In this chapter, we focus on tiny WSN nodes and study algorithms that make minimal assumptions on the routing layer, and that operate using the most basic version of contention-based medium access without tight synchronization of node clocks and communication schedules. Specifically, in our peer-to-peer optimization algorithms nodes only communicate with their immediate neighbors and *no multihop routing is required or exploited*. Within this class of solutions, energy efficiency can be achieved by reducing the actual time required for the algorithm to converge. A first-order approximation is simply the classical convergence rate in terms of expected progress per iteration. However, since different methods dictate different communication patters, the problem is more complex. To increase efficiency, one can either try to exploit the broadcast nature of the wireless medium (to avoid repeated unicast transmissions with the same information) or limit the number of retransmissions due to collisions.

6.3.2 MAC Layer

The wireless communication channel in a WSN is a resource that all nodes share. For proper operation of the network, it is therefore critical that some efficient channel allocation scheme is used to control access to this shared resource. Such schemes or protocols usually reside (logically) in the so-called Medium Access Control (MAC) layer, and we will now touch upon some of the characteristics of the MAC layer relevant for our purposes.

The target application is sensor networks, and the dominating standard for such networks is the IEEE standard 802.15.4 (IEEE, 2006). This protocol has support for several modes of operation. Since the algorithms we consider are based on peer-to-peer communication, we will focus on 802.15.4 in peer-to-peer operation. In this mode of operation, 802.15.4 uses so-called unslotted Carrier Sense Multiple Access - Collision Avoidance (CSMA-CA); see, e.g., Rom and Sidi (1990, Chapter 4) for details on CSMA. Unslotted CSMA-CA basically works like this: a node that wishes to transmit listens to the channel. If the channel appears to be idle, the node starts to transmit. On the other hand, if the channel is not idle, the node backs off according to the scheme presented in Fig. 6.3. In the case that two or several nodes' transmissions overlap and they are so close to each other that they cause sufficiently high interference, then the nodes immediately stop transmitting and wait for a random time before transmitting again.

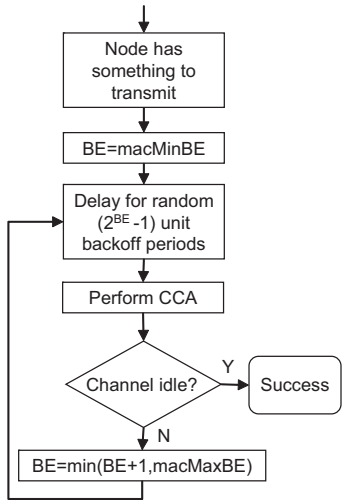


Figure 6.3: Flowchart showing the MAC scheme, with the following abbreviations: Clear Channel Assessment (CCA) and Backoff Exponent (BE).

Some of the optimization algorithms that we will implement are synchronized in the sense that they require that all communication activities have been completed in order to proceed to the next step in the algorithm. Thus, we need reliable communications, and we can address this by using acknowledgement packets (ACKs). There is an ACK functionality in 802.15.4, but it is not always suited for our purposes. More specifically, we will have problems with acknowledging broadcast messages, since the built-in ACK functionality directly replies with an ACK without any random waiting time. The direct reply could lead to many collisions if the broadcast message is received and acknowledged by several nodes at the same time. To solve this predicament, we will have some additional functionality at the application layer; see Section 6.5.1.

6.4 Algorithms

We will now have a detailed look at the algorithms that we will implement. For all algorithms, we make, as usual in this thesis, the following assumptions.

Assumption 6.4.1. *i) The nodes form a connected network. ii) The functions f_i are convex and the set $\mathcal{X}$ is convex and closed.*

6.4.1 Dual Decomposition Based Algorithms

Dual based methods introduce one "local" copy of x in each node, and update these estimates so that they converge towards the optimum. Formally, this is done by introducing multiple copies of the global variables, requiring that these should be equal, and then relaxing these constraints; see Rabbat et al. (2005).

We re-write the problem as

$$\begin{array}{ll} \underset{(x_1,\dots x_n)}{\text{minimize}} & \sum_{i=1}^n f_i(x_i) \\ \text{subject to} & x_i \in \mathcal{X}, \quad i = 1, ..., n \\ & x_i = x_j, \quad (i,j) \in \mathcal{E}. \end{array} \tag{6.2}$$

If the communication topology is connected, then the formulation (6.2) is equivalent to (6.1). In general, all these constraints are not necessary: a necessary and sufficient condition for equivalence between the two formulations is that equalities are introduced for the smallest set of links that keep the nodes connected (cf. Schizas et al. (2007)). We can re-write the problem into a more compact form

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \sum_{i=1}^n f_i(x_i) \\ \text{subject to} & x_i \in \mathcal{X}, \quad i = 1, ..., n \\ & Ax = 0, \end{array} \tag{6.3}$$

where $x = \begin{pmatrix} x_1 & \dots & x_n \end{pmatrix}^\intercal$, and $A \in \mathbb{R}^{|\mathcal{E}| \times n}$. Each row in the matrix A corresponds to a link in $\mathcal{G}$, and the elements of A are defined as (in the literature, this type of matrix is sometimes denoted the directed incidence matrix)

$$A_{li} = \begin{cases} 1, & \text{if link } l \text{ connects node } i \text{ with node } j,\ i > j \\ -1, & \text{if link } l \text{ connects node } i \text{ with node } j,\ i < j \\ 0, & \text{otherwise.} \end{cases} \tag{6.4}$$

We can of course also write the previously mentioned minimum needed number of equalities in this way.

In this formulation, the computations are complicated by the constraint $Ax = 0$, which couples the local estimates of the network-wide decision variables across nodes. To distribute the computations to the nodes, we apply Lagrange dual decomposition to the formulation (6.3). Formally, we introduce the Lagrangian

$$\mathrm{L}(\lambda, x) = \sum_{i=1}^n f_i(x_i) + \lambda^\intercal A x$$

and the dual function

$$\begin{array}{rll} \mathrm{d}(\lambda) = & \min\limits_{x} & \mathrm{L}(\lambda, x) \\ & \text{subject to} & x_i \in \mathcal{X}, \quad i = 1, ..., n. \end{array} \tag{6.5}$$

The Lagrange multipliers, λ, can be interpreted as "prices" for disagreeing with neighbors on the best global variable value, as previously discussed.

We make the following additional assumption.

Assumption 6.4.2. *i) Strong duality holds for* (6.3)*. ii) the subgradients* $a(\lambda)$ *of* $\mathrm{d}(\lambda)$ *are bounded, with* $\|a(\lambda)\| \leq \varphi$ *for all values of* λ*.*

The dual function can be written as $\mathrm{d}(\lambda) = \sum_{i=1}^{n} \mathrm{d}_i(\lambda)$ with

$$\begin{aligned} \mathrm{d}_i(\lambda) = \quad & \min_{x_i} \quad f_i(x_i) + x_i[A^{\mathsf{T}}\lambda]_i \\ & \text{subject to} \quad x_i \in \mathcal{X}. \end{aligned} \tag{6.6}$$

Note that the dual function is separable, i.e., for fixed λ the nodes can individually decide the optimal values of its local variables x_i by evaluating $\mathrm{d}_i(\lambda)$. When strong duality holds (which it does by Assumption 6.4.2; also, see Section 2.7.1) the optimal value of (6.1) is the same as the optimal value of the Lagrange dual problem

$$\underset{\lambda}{\text{maximize}} \quad \sum_{i=1}^{n} \mathrm{d}_i(\lambda). \tag{6.7}$$

However, when the cost functions are not strictly convex, there may be problems recovering the primal optimal $x^\star$; see, e.g., Example 2.7.1.

Subgradient Algorithm

The dual function is not differentiable in general, and the simplest way to perform the maximization in (6.7), is to use a subgradient algorithm. Thus, we need a subgradient, a, of d at λ. One subgradient is given by

$$a = Ax^\star(\lambda), \tag{6.8}$$

where $x^\star$ is the x that minimizes the Lagrangian for the given λ (see, e.g., Bertsekas et al. (2003, Chapter 8) for details). Now we have all the pieces needed for a decentralized algorithm that solves (6.3), and the result can be seen in Algorithm 11.

Under Assumptions 6.4.1 and 6.4.2, and since we are using a fixed stepsize and the dual function is not necessarily differentiable, we can only guarantee that the best iteration of Algorithm 11 can be shown to converge to a ball around the optimal value $f^\star$; see Section 2.6.1. There are also convergence results for this algorithm when there are packet losses and the cost function is quadratic; see Rabbat et al. (2005).

As we previously pointed out, the dual function is in general non-differentiable due to the lack of strict convexity. This means that it can be troublesome to find the optimal $x^\star$ (see Example 2.7.1) and it also hinders us from using other methods than those based on subgradients. One remedy is to add a quadratic term to the

Algorithm 11 Dual Based Subgradient Method (DBSM) .

1: Let $k \leftarrow 0$ and $\lambda^{(0)} \leftarrow 0$.
2: **loop**
3: Each node i solves

$$\begin{array}{ll} \underset{x_i}{\text{minimize}} & f_i(x_i) + x_i \left(\sum_{\{l|A_{li}\neq 0\}} A_{li}\lambda_l^{(k)} \right) \\ \text{subject to} & x_i \in \mathcal{X} \end{array}$$

to get $x_i^{(k)}$.
4: Each node i transmits its optimal $x_i^{(k)}$ to its neighbors.
5: Each node i updates the elements of $\lambda^{(k)}$ that are part of its Lagrangian

$$\lambda_l^{(k+1)} \leftarrow \lambda_l^{(k)} + \alpha \sum_{\{j|A_{lj}\neq 0\}} A_{lj} x_l^{(k)},\ l \in \{m|A_{mi} \neq 0\}.$$

6: $k \leftarrow k + 1$.
7: **end loop**

Lagrangian, and we form the so-called *Augmented Lagrangian*

$$\mathrm{L}_c(\lambda, x) = \sum_{i=1}^{n} f_i(x_i) + \lambda^{\intercal} A x - \frac{c}{2} ||Ax||_2^2.$$

Using the Augmented Lagrangian, we can use the method of multipliers or the alternating direction multiplier method (Bertsekas and Tsitsiklis, 1997). The most interesting theoretical property of these algorithms is that they converge for all constant stepsizes. However, in practice, tuning is still crucial to give acceptable performance, and we will not investigate these variants in this thesis.

6.4.2 Primal Subgradient Algorithms

The functions f_i are in general non-differentiable. Therefore, we have to resort to using subgradient based algorithms. Let each (convex) component f_i have a subgradient $a_i(x)$ at x and we assume that they are bounded.

Assumption 6.4.3. *The subgradients $a_i(x) \in \partial f_i(x)$ of f_i are bounded, $\|a_i(x)\| \leq \varphi$, for all values of $x \in \mathcal{X}$.*

The simplest algorithm that uses the primal formulation (6.1) and that is suitable for distributed implementation is the *incremental subgradient algorithm*. The basic version is based on estimate passing in a logical ring (Rabbat and Nowak, 2004), but it does not fulfill our assumptions of a peer-to-peer algorithm, as we have previously discussed. We will instead use our novel MISM from Chapter 3, which is a peer-to-peer algorithm. The algorithm is summarized in pseudocode in Algorithm 12.

Algorithm 12 Markov Incremental Subgradient Method (MISM) .

1: Initialize x_0 and α. Let $k \leftarrow 0$ and $w_k \leftarrow 1$.
2: **loop**
3: At node w_k, compute a subgradient, a_{w_k}, for $f_{w_k}(x^{(k)})$.
4: $x^{(k+1)} \leftarrow \mathcal{P}_{\mathcal{X}}\{x^{(k)} - \alpha a_{w_k}\}$.
5: Send $x^{(k+1)}$ to a random neighbor, $w^{(k+1)}$, with transition probability according to P.
6: $k \leftarrow k + 1$.
7: **end loop**

Algorithm 13 Linear Iteration Method (LIM) .

1: Initialize $x_i^{(0)} \leftarrow c_i$ for all $i = 1, ..., n$. Set $k := 0$.
2: **loop**
3: Each node i broadcasts $x_i^{(k)}$ to all of its neighbors.
4: Each node i updates its x according to $x_i^{(k+1)} \leftarrow W_{ii}x_i^{(k)} + \sum_{j\in\mathcal{N}(i)} W_{ij}x_j^{(k)}$.
5: $k \leftarrow k + 1$.
6: **end loop**

6.4.3 Linear Iterations for the Quadratic Case

For the special case with quadratic cost functions

$$f_i(x) = \frac{1}{2}(x - c_i)^2,\ i = 1, ..., n, \tag{6.9}$$

the optimal solution is the average, $x^\star = \sum_{i=1}^n c_i/n$, and any of the vast number of algorithms for distributed averaging can be used. As discussed in Chapter 5, the simplest algorithms have the following form

$$x^{(k+1)} = Wx^{(k)},$$

where W is a weighting matrix. Necessary and sufficient conditions on W to guarantee convergence to the average have been given in Section 2.3, where we also provide simple heuristics that can be used to devise W using only local topology information. Furthermore, in Section 5.2, we have discussed ways to optimize the convergence rate of this iteration. The resulting algorithm is summarized in Algorithm 13.

6.5 Simulations

We have concentrated our efforts on evaluating the previously introduced algorithms: DBSM (Algorithm 11), MISM (Algorithm 12), and LIM (Algorithm 13). There are, as mentioned before, an abundance of algorithms, and we have chosen

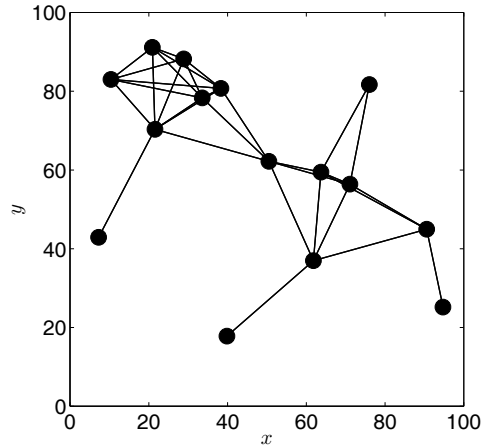


Figure 6.4: The topology for one of the random networks with 15 nodes used in the Monte Carlos simulations. The filled circles denote nodes and a line between two nodes indicate that they can transmit to each other.

these three due to their interesting properties and their simplicity. We consider the standard example of least-squares estimation: the nodes should minimize the deviation between the estimate and the measurements in the quadratic sense; see (6.9).

We now proceed with describing the NS2 simulation setup and implementation details of the algorithms.

6.5.1 NS2 Implementation

NS2 is an event driven network simulator that is widely used in the networking community. We decided to use it for our simulations, since it provides detailed and validated descriptions of the MAC layer and networking algorithms, including an 802.15.4 module and several models for wireless radio propagation. We have modified the simulation environment, such that the behavior of the nodes at the application layer is consistent with the optimization algorithms described in this thesis, giving us the possibility to easily change the key parameters of each algorithm through high level scripting (using the OTcl interface). We ran our application over the 802.15.4 module for NS2 (Zheng and Lee, 2006), and the setup is summarized in Table 6.1. The 802.15.4 module was set to peer-to-peer asynchronous mode, without any beacon (the beacon, when active, is used to synchronize the network). To model the wireless channel, we use the two-ray-ground propagation model, which considers both the direct path and the ground reflection path, without any shadowing or fading. For further information about the simulations, see Carretti (2008).

Table 6.1: NS2 simulation setup.

Parameter	Setting
Physical and MAC layer	802.15.4
Transmit Power	0.282 W
Antenna	Omnidirectional at 1.4 m above the ground
Frequency	2.4 GHz
Propagation Model	Two-ray-ground
Queue	Droptail FIFO

6.5.2 DBSM

Due to the dual based subgradient algorithm's resilience to packet losses (Rabbat et al., 2005), we use unreliable broadcast. This means that each node broadcasts its estimate to its neighbors but does *not* wait for ACKs. However, each node will wait for a specific *waiting time* to receive estimates from its neighbors. During this waiting time, each node broadcasts its estimate ζ times to the neighboring nodes, to increase the probability of successful transmissions. The waiting time and the number of retransmissions during the waiting time are tunable parameters. Finally, the DBSM also has a tunable stepsize parameter, α. With a fixed stepsize, complete convergence is not guaranteed, and we have to be satisfied with convergence to a ball around the optimal solution; see Section 2.6.1.

6.5.3 MISM

The MISM has two tunable parameters, namely the stepsize α and the transition probability matrix P (this could be considered to be many parameters). P can be tuned to increase the exploration rate of the network. However, it is hard to perform this tuning in a distributed fashion. Therefore, we use the simple scheme given by (2.9).

6.5.4 LIM

For the linear iteration algorithm, we use reliable broadcast, since it is crucial that the correct values are used in each iteration. This is due to the fact that in the nominal case, the average is kept constant over time, and this average is redistributed over the network to yield a uniform distribution. If some values are missing, this value will disappear from the average and the algorithm will converge to the wrong point. This is the same mechanism that makes this type of iteration sensitive to noise (which we do not consider). The problem comes from the 1-

eigenvalue corresponding to the eigenvector with all elements equal to one (this is the eigenvalue corresponding to consensus).

We implement reliable broadcast by letting each node broadcasts its current estimate to its neighbors, and then wait for ACKs. If the node does not get an ACK within a fixed time-out window, it re-broadcasts its estimate again with a list of nodes from which it did not get any ACKs. The time-out window length is a tunable parameter and it is reasonable to let it depend on the number of nodes in the network (whenever this is known).

6.5.5 Performance Metrics

Depending on the precise configuration, most current WSN nodes have higher power consumption when receiving than when transmitting. For example, the Tmote Sky mote typically consumes 19.7 mA when receiving, and it consumes 17.4 mA when the radio is transmitting at maximum power 0 dBm = 1 mW (Tmote, 2008). This is sometimes neglected in the current literature, where listening is considered to be practically free or at least very cheap. In general, this may be true; transmitting can in fact be much more expensive than receiving, but it is usually not the case in WSNs.

Since the nodes are active all the time, due to the peer-to-peer mode, the relevant metric for nodes with higher power consumption while receiving than transmitting is the rate of convergence in terms of time. This is the metric we use in this chapter. On the other hand, in a setup where transmission is much more expensive than reception, then the relevant metric could be convergence time in terms of number of packets sent.

6.5.6 NS2 Simulations

For DBSM, we set the waiting time to 0.1 seconds and the number of retransmissions, ζ, to 3. These parameters were found by direct experimentation. Furthermore, we set the stepsize α to 0.01, 0.04, and 0.08. For MISM, we set the stepsize α to 0.01 and 0.08. For LIM, we set the time-out window to $1 + n/20$ seconds, which is a value that performed well in simulations.

We imported Matlab generated random network topologies into NS2. The graphs were generated as follows: First, we randomly placed the nodes according to a uniform distribution on a square surface. Second, each node was assumed to be connected with all nodes within a radius r, which is a so-called unit disk graph. The radius r was initially set to be the maximum distance between a pair of nodes, which implies that the network was fully connected at the start. Finally, the radius was gradually decreased just until the network ceased to be connected. We then configured the radio environment in NS2 such that each node had exactly those neighbors specified by the imported graph.

The random network topologies were used to evaluate typical performance of the algorithms, and we performed 1000 Monte Carlo simulations for each setup.

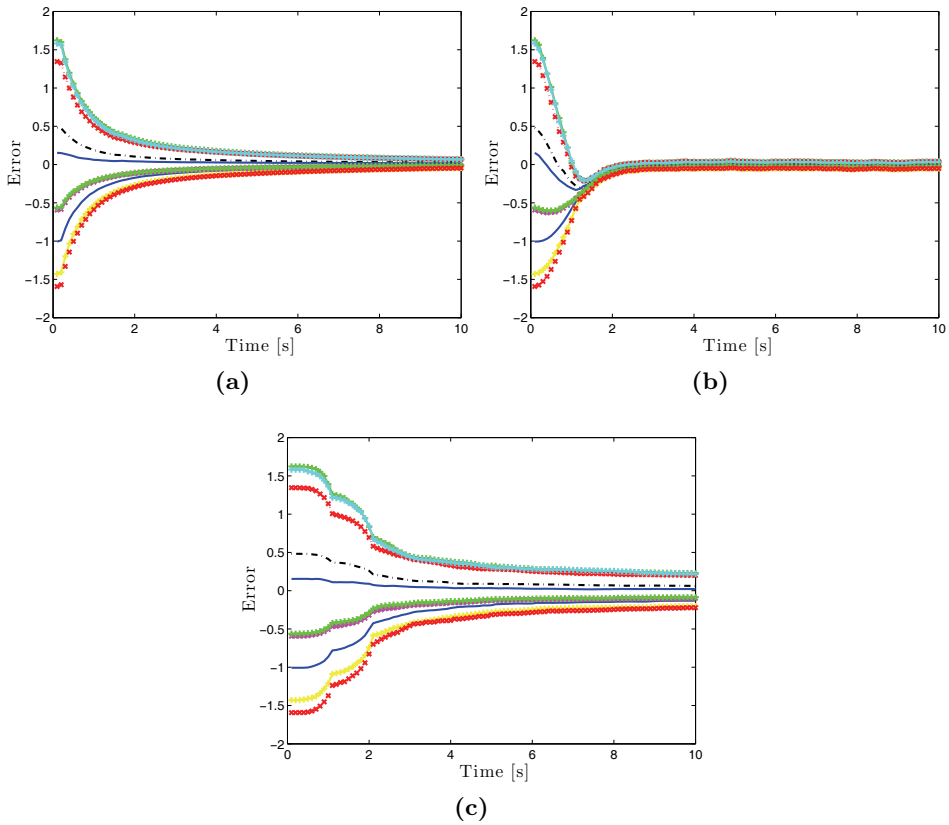


Figure 6.5: These plots show the *average* of 1000 Monte Carlo simulations of the individual node error for 10 node random topologies. (a) DBSM, with $\alpha = 0.08$. (b) MISM, with $\alpha = 0.01$ (c) LIM.

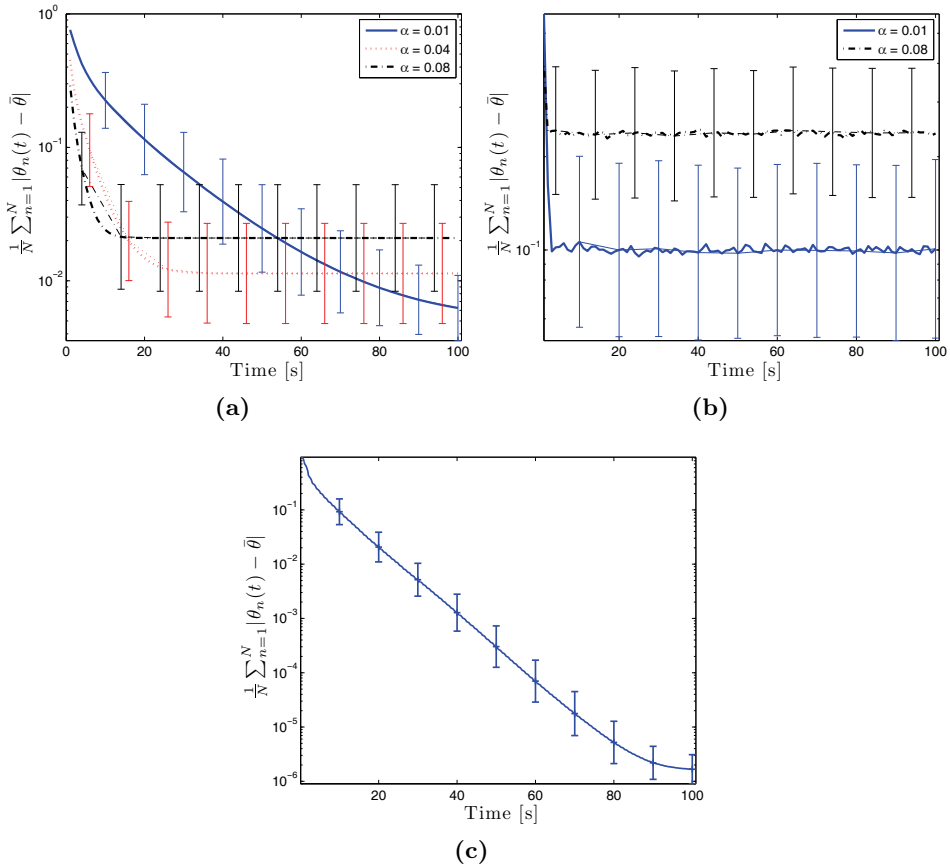


Figure 6.6: These plots show the *average* of 1000 Monte Carlo simulations of the average of the absolute error for all nodes versus time for the 5-node random topology; see Fig. 6.4 for an example of a typical random network. The thick lines are the averages and the error bars show the standard deviation[1]. The optimal x is $x^\star = \bar{x} = 2.04$. (a) DBSM. The algorithm was run with 3 different α values according to the legend. (b) MISM. The algorithm was run with 2 different α values according to the legend.(c) LIM.

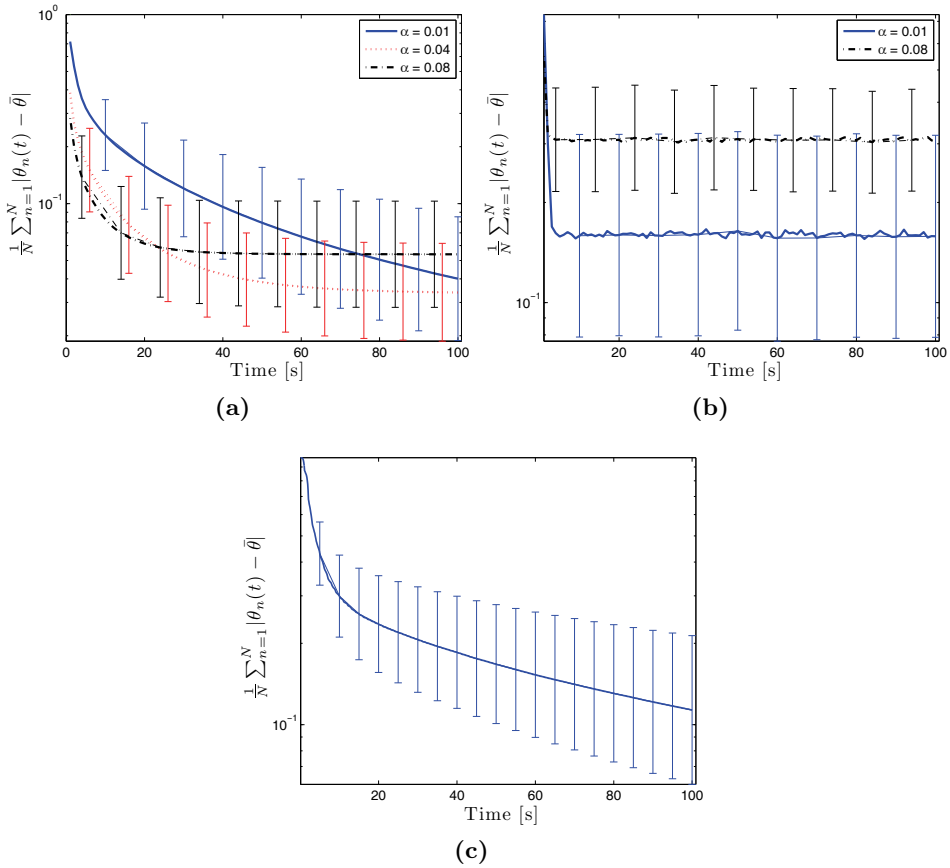


Figure 6.7: These plots show the *average* of 1000 Monte Carlo simulations of the average of the absolute error for all nodes versus time for the 15-node random topology; see Fig. 6.4. The thick lines are the averages and the error bars show the standard deviation[1]. The optimal x is $x^\star = \bar{x} = 1.94$. (a) DBSM. The algorithm was run with 3 different α values according to the legend. (b) MISM. The algorithm was run with 2 different α values according to the legend.(c) LIM.

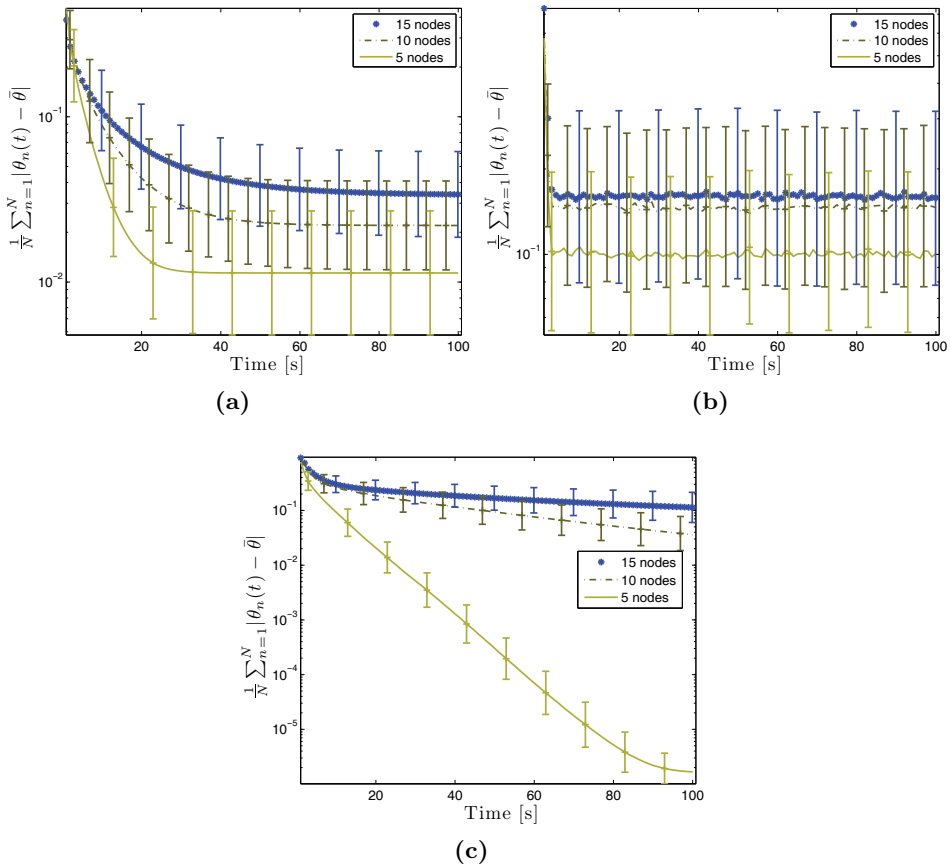


Figure 6.8: These plots show the *average* of 1000 Monte Carlo simulations of the average of the absolute error for all nodes versus time for random topologies with 5, 10, and 15 nodes. The thick lines are the averages and the error bars show the standard deviation[1]. (a) DBSM. The algorithm was run with $\alpha = 0.04$. (b) MISM. The algorithm was run with $\alpha = 0.01$. (c) LIM.

We started with 10-node networks with random positions. The number of nodes is low enough to admit a detailed inspection of the algorithm behavior, but still high enough in order for the algorithms to have interesting behavior. The individual node errors are shown in Fig. 6.5. We can see that the MISM has the fastest convergence to stationarity, but with higher bias than the DBSM (almost no bias). The LIM should theoretically not have any bias, but convergence is probably hindered by quantization. Furthermore, the LIM continues to improve after 10 seconds (the time span 0 - 10 seconds is shown), as can be seen in Fig. 6.8; however, the LIM does not converge completely.

Then, we simulated 5-node networks with random positions; see Fig. 6.6. The average absolute errors are shown in Fig. 6.7. Again, we can see that MISM has the fastest convergence to stationarity but has larger bias than the DBSM and the LIM. The LIM is the slowest method, but it reaches high accuracy in the end.

Next, we simulated 15-node networks with random positions; see Fig. 6.4. The average absolute errors are shown in Fig. 6.7. Again, we can see that the MISM has the fastest convergence to stationarity but has larger bias than the DBSM and the LIM. This time, the DBSM comes closer to the optimal value than the LIM.

Finally, we compared, as illustrated in Fig. 6.8, the three algorithms on random networks with 5, 10, and 15 nodes, using the best parameters found (DBSM: $\alpha = 0.04$. MISM: $\alpha = 0.01$). The plots indicate that LIM has much slower convergence when the number of nodes is increased, and that LIM does not seem to scale well with the number of nodes. DBSM is less affected by the number of nodes, but the convergence time is slower and the bias is increased when the number of nodes is increased. MISM is least affected by the number of nodes, and the convergence behavior is almost the same when the number of nodes is increased. However, the bias will also increase.

6.6 Implementation on Tmote Sky Motes

In this section, we describe how the algorithms were implemented on the Tmote Sky motes using the open-source operating system TinyOS 2.0.1 (TinyOS, 2008). The implementation is basically the same as in the NS2 simulations, but there are some real world challenges.

6.6.1 Implementation Aspects

We implemented the algorithms on TMote Sky motes with TinyOs 2.0.1. The Tmote Sky mote is a low power wireless node equipped with a 250 kbps 2.4 GHz IEEE 802.15.4-compliant transceiver; see Tmote (2008) for more details.

[1] If the error bars are plotted using $\bar{x} \pm \sigma$, i.e., average $\pm$ standard deviation, then the lower value, $\bar{x} - \sigma$, may end up below zero. Negative values will ruin the plot due to the log scale. To get a more visually pleasing plot, the upper value of the error bar is $u = \bar{x} + \sigma$, while the lower value of the error bar is $l = \frac{\bar{x}}{1+\frac{\sigma}{\bar{x}}}$, and the following holds $\bar{x} = \sqrt{l \cdot u}$.

We artificially limit the transmission power to -15 dBm $= 31.6$ μW to obtain interesting network topologies in our indoor environment. The resulting graphs are *not* fully connected, which are more interesting than fully connected graphs. The implementation consists of three distinct phases: first, the motes enter a startup phase with neighbor discovery. Second, the motes execute the chosen algorithm. Third, the trace of the execution, for each mote, is downloaded via a base station.

Neighbor Discovery

In the first phase, the motes identify neighbors with which it can sustain reliable communications. This step is necessary in order to know where the packets should be sent. The hard part is that the algorithms need symmetric graphs, otherwise they will not function properly. Thus, each mote needs to figure out if the motes that it can hear also can hear it. However, note that this is a consensus type problem, and if the links are unreliable, it is in fact impossible to agree with complete accuracy (Lynch, 1996, Theorem 5.1).

Neighbor discovery is not part of the optimization algorithms per se, and therefore, we implemented a crude and simple neighbor discovery algorithm. In this scheme, each mote broadcasts HELLO packets in a random non-periodic fashion. If a mote receives a HELLO packet, it sends an ACK indicating that it can hear the mote that sent the HELLO packet. To diminish the influence of stochastic effects, we program the motes to discard packets with a low Received Signal Strength Indicator (RSSI) value and a low Link Quality Indicator (LQI) value. The RSSI value is an indicator of the power of the received signal. If there is no interference from other devices this can be very useful. However, there can still be many reception errors even with a high RSSI value if the power comes from interfering transmissions. Therefore, we also require the LQI value to be above a certain threshold, since the LQI value is a measure of the number of errors of the received symbols defining each packet. This is not a perfect solution, but it is sufficient for our purposes and we note that link estimation is inherently tricky (see, e.g., Srinivasan and Levis (2006) for some experiments and discussion). Furthermore, the motes also keep track of the ratio between the number of received ACKs and the number of HELLO packets sent. Finally, if the ratio is above a certain threshold, the corresponding mote is considered to be a neighbor where packets can be reliably sent.

Algorithm Execution

The algorithms are implemented essentially in the same way as in the NS2 implementation. The greatest difference is that all data sent between motes in the real WSN have to be converted to integer values. This will make sure that the software is portable to other hardware, since the standard on how to store variables in memory is hardware dependent. In our implementation, we use 4 digits accuracy to transmit the iterate between the motes and we loose some precision in this process, which will show up as quantization effects and noise.

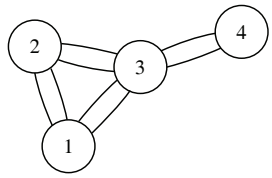


Figure 6.9: This is the topology used in the experiments with 4 nodes. The figure only illustrates the logical topology, i.e., neither the actual distance between the nodes nor the nodes' positions can be seen. Each solid line indicates a directed link.

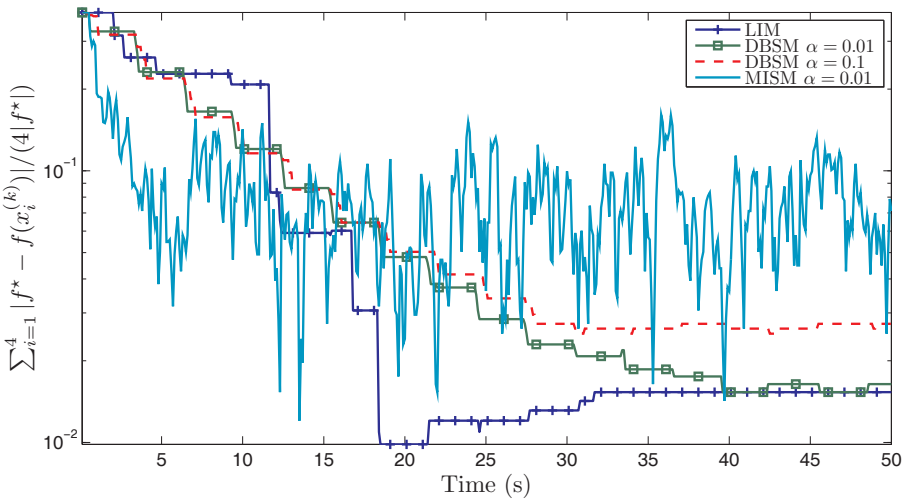


Figure 6.10: Convergence results with 4 nodes.

Download Trace

In the download phase, the base station queries each mote. Then, the motes, one at a time, transmit the trace data at full transmission power (1 mW) to the base station.

6.6.2 Experimental Results on Tmote Sky Motes

We performed experiments using a 4 mote and a 6 mote configuration. Let us start with the 4 mote case. The logical topology is shown in Fig. 6.9. The figure only shows which links the motes used, i.e., there is not a direct correspondence with the physical location of the motes. The convergence of the three algorithms for different stepsizes are shown in Fig. 6.10. The MISM has rapid initial convergence, but after

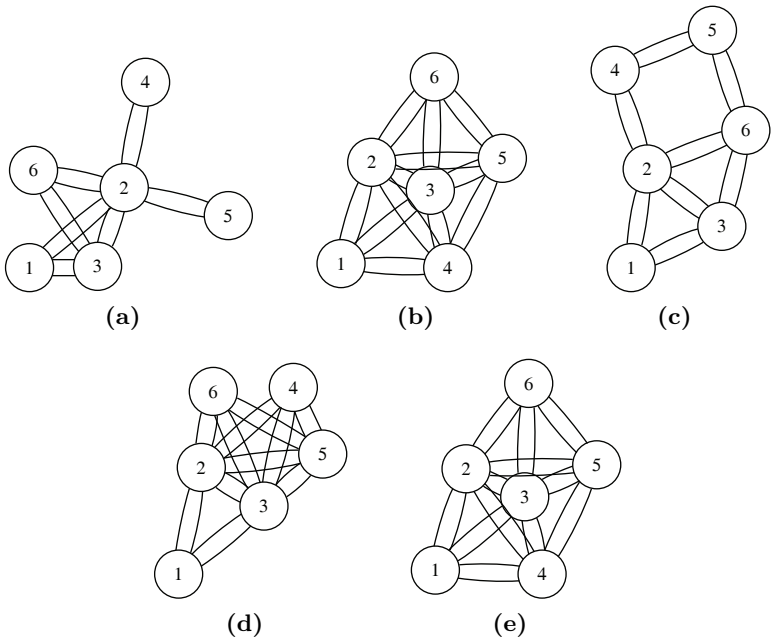


Figure 6.11: These are the topologies used in the experiments with 6 nodes. Due to the stochastic nature of the wireless channel and the communication protocols, the neighbor discovery phase, which is run before each execution phase, did not result in identical topologies. The figure only illustrates the logical topology, i.e., neither the actual distance between the nodes nor the nodes' positions can be seen. Each solid line indicates a directed link. a) LIM. b) DBSM with $\alpha = 0.01$. c) DBSM with $\alpha = 0.1$. d) MISM with $\alpha = 0.01$. e) MISM with $\alpha = 0.1$.

5 seconds it seems to fluctuate around a plateau. The MISM was also executed with $\alpha = 0.1$, but since its convergence plot bears a very strong resemblance with the plot for $\alpha = 0.01$, it is not included in the figure to increase the readability of the plot. The DBSM converges steadily for both $\alpha = 0.1$ and $\alpha = 0.01$, but it has slower initial convergence behavior compared to the MISM. The LIM performs similarly to the DBSM in the beginning, but it shows a sudden drop after 15 seconds. In this scenario, the LIM outperforms the other methods. If a rapid initial convergence is desired, the MISM could be useful. Also note that we should not expect convergence to the true optimal point due to quantization effects in the communications.

We also performed experiments using a 6 mote configuration. The logical topology is shown in Fig. 6.11. Due to the fact that the neighbor discovery phase is executed before the execution of each algorithm and the fact that the wireless channel is stochastic in combination with a simplistic neighbor discovery algorithm, the topologies are not the same. The figure only shows which links the motes used, i.e., there is not a direct correspondence with the physical location of the motes. The

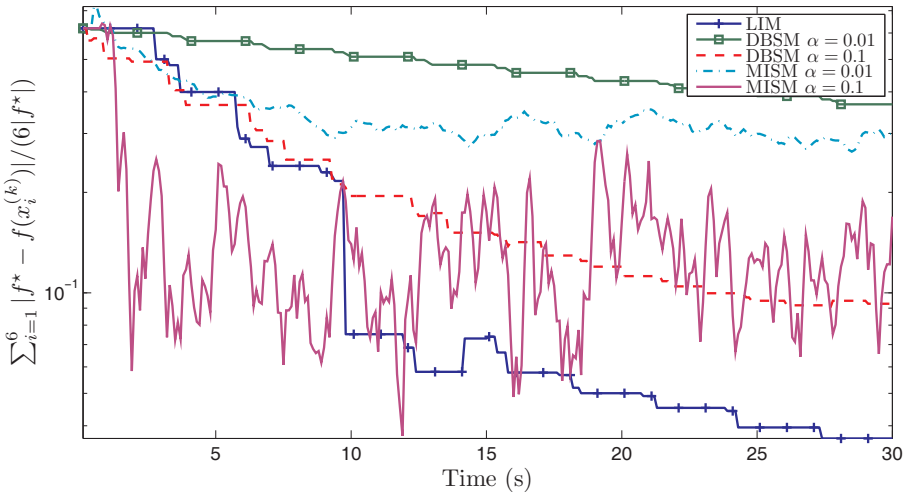


Figure 6.12: Results with 6 nodes.

convergence of the three algorithms for different stepsizes are shown in Fig. 6.12. The MISM with $\alpha = 0.1$ has rapid initial convergence, but after 3 seconds it starts to fluctuate around a constant level. The MISM with $\alpha = 0.01$ converges at the same pace as the DBSM with $\alpha = 0.1$ and the LIM in the start, but after circa 10 seconds it also levels off. The DBSM with $\alpha = 0.1$ converges steadily close to the LIM in the beginning. The DBSM with $\alpha = 0.01$ is clearly outperformed by the other algorithms, even though it shows a constant, but slow, convergence rate. Finally, the LIM's convergence behavior is very similar to the other methods in the beginning and it shows a sudden drop at around 10 seconds. The LIM is the best algorithm also in this experiment if accuracy is the first priority. The MISM can be useful in applications where only swift initial convergence is needed. In addition, in Fig. 6.13, we take a detailed look at the individual values of the motes for the 6 mote experiment. We see that the characteristics of each algorithm are consistent with the characteristics in the simulations in Fig. 6.5, although the simulations plots are averages over 1000 Monte Carlo simulations (also note that the time scale is different). This strengthens our faith in the simulations. Furthermore, in the MISM, the individual mote values quickly converge to a time varying signal. The LIM converge quite fast to the true optimum, while the DBSM can be seen to take a long time before the mote values converge to the same value.

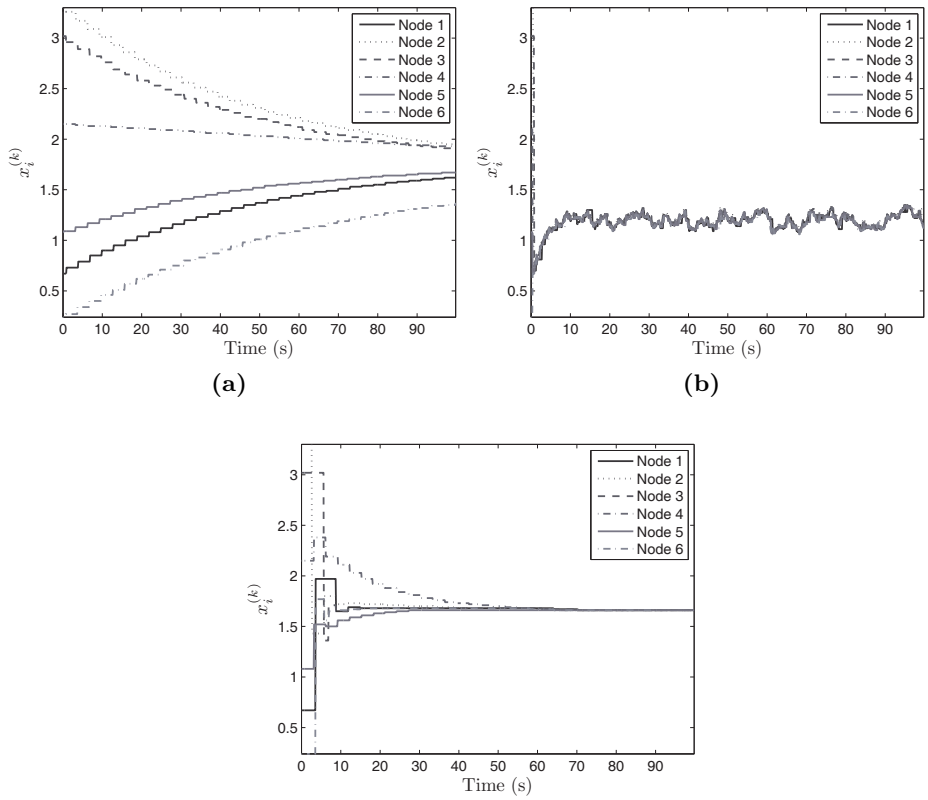


Figure 6.13: These plots show the individual value for each mote in the 6-node networks shown in Fig. 6.11. (a) DBSM. The algorithm was run with $\alpha = 0.01$. (b) MISM. The algorithm was run with $\alpha = 0.01$. (c) LIM.

6.7 Summary

In this chapter, we have considered peer-to-peer algorithms for distributed optimization in wireless sensor networks. We have evaluated the performance of three specific algorithms on network nodes equipped with 802.15.4 compliant radios. While the linear iteration algorithm is tailored for averaging (least-squares) problems, where it excels in networks with few nodes, it is not applicable to general optimization problems. The other two algorithms are complementary and work surprisingly well in our simulations.

The simulations indicate that broadcast algorithms, where each node needs data from all of its neighbors to complete one iteration, do not scale with the number of nodes. One way of mitigating this is to divide the network into clusters and run the broadcast algorithm within each cluster; see, e.g., Son et al. (2005).

An interesting observation is that it is a design choice to put the stochastic behavior of the resulting system (optimization algorithm plus communication mechanism) either in the algorithm or in the communication mechanism. If we use a stochastic algorithm, in which the updates are explicitly stochastic, then we can use an unreliable communication mechanism with known time delay and number of required packets. On the other hand, if we use a deterministic algorithm, then we need a reliable communication mechanism, in which the time delay and number of packets needed are uncertain.

The algorithms require the graph to be symmetric, which turned out to be a bit problematic to guarantee with the simplistic neighbor discovery algorithm we used. Neighbor discovery is an active area of research (see, e.g., Hamida et al. (2006) and Dutta and Culler (2008)) and the performance could probably be increased if we use a protocol available in the literature. Furthermore, the problem could be mitigated with an extension of the simplistic neighbor discovery algorithm, which should ensure that the resulting graph is symmetric with high probability. Another potential remedy is to extend the optimization algorithms to handle unsymmetric graphs as well. We did not have this problem in the NS2 simulations, since we specified the radio ranges exactly to fit the generated graph.

The real WSN experiments show that the algorithms work rather well also in a real WSN. However, since most of the algorithms are quite stochastic in their nature, more experiments need to be conducted in order for us to say something with more accuracy. It is also interesting that the algorithm choice affects the communication pattern, which in turn will change the communication topology. That is, links that are considered reliable under one specific optimization algorithm may be totally unreliable when another optimization algorithm with a different communication pattern is used, due to interference.

Future work includes evaluating alternative algorithms, and investigating, in greater detail, the impact of tuning of the parameters.

Chapter 7

Discussion and Future Work

In this thesis, we have developed three novel optimization algorithms that only rely on peer-to-peer communication and are suitable to be used in networked systems. Furthermore, we have designed resource allocation protocols for FDMA and STDMA communication networks using decomposition techniques. We have also devised a distributed negotiation scheme for optimal rendezvous points for multi-agent systems, as well as we have looked into the problem of boosting the convergence of linear iterations for distributed averaging. Finally, we have implemented and evaluated some of the most promising optimization algorithms on our WSN testbed.

In this chapter, we summarize and discuss each chapter of the thesis. In addition, we also outline possible and natural extensions, as well as broader ideas for future work.

7.1 Chapter 3

In Chapter 3, we have presented three optimization algorithms where there is no need for a central processing unit to whom each node should communicate its local subgradient. Two of the algorithms solve a general convex non-smooth optimization problem with an additive objective function, and the third algorithm solves a non-smooth separable resource allocation problem with a constant sum constraint.

First, we have proposed a novel randomized incremental subgradient method, the Markov incremental subgradient method, which is well suited for decentralized implementation in networked systems. The algorithm is a generalization of the deterministic incremental subgradient method and the randomized incremental subgradient method due to Nedić and Bertsekas. These two algorithms can be recovered by choosing the transition probability matrix of the Markov chain in a special way. The algorithm has been analyzed in detail with a proof of convergence proof and a bound on the expected number of needed iterations to reach an *a priori* specified accuracy.

Second, we have described an iterative subgradient-based method for solving

coupled optimization problems in a distributed way given restrictions on the communication topology. In order to allow great flexibility in the information exchange architecture and distribute calculations, we combined the local subgradient updates with a consensus process. This means that computing nodes can work in parallel with each other and use localized information exchange. For analysis purposes, we used results from consensus theory and employed approximate subgradient methods to study convergence properties of the proposed scheme. A connection is established between the number of consensus steps and the resulting level of optimality obtained by the subgradient updates.

Third, we have shown that under some mild technical assumptions, the center free algorithm introduced by Ho et al. (1980) can be extended to the subgradient case. Furthermore, the algorithm converges in an ε-sense if the stepsize is chosen sufficiently small and the number of consensus iteration per subgradient update is chosen sufficiently large.

7.1.1 Future Work

Subgradient methods are very applicable due to their simplicity; the drawback is that they can be slow. It would therefore be interesting to see if some type of higher order methods could be used, which could potentially lead to faster convergence.

Different choices of the stepsize should be explored, e.g., diminishing stepsizes. Furthermore, techniques from stochastic optimization (see, e.g., Kushner and Yin (2003)) could potentially be used. However, for the MISM algorithm, this is already done to some extent in Ram et al. (2008).

How should the different parameters be tuned? Should we decrease the stepsize or increase the number of consensus iterations to get closer to optimality? This is a variant of the interesting trade-off: is it beneficial to communicate more information per iteration and use a faster converging algorithm? The experience from this thesis indicate that the choice will very much depend on the specific application and setup, which is probably not that surprising.

To improve scalability, it is the author's intuition that the network should be partitioned into clusters and run some algorithm in each cluster, then have a master problem on top; see, e.g., Son et al. (2005).

It would be useful to consider asynchronous or partially asynchronous communications, perhaps using the framework presented in Bertsekas and Tsitsiklis (1997). As mentioned in Section 3.4, this has already been done to some extent in Nedić and Ozdaglar (2007*b*). Furthermore, if the methods are going to be used in embedded systems in applications, the implementation should be made robust against model errors, quantization, and packet drops.

7.2 Chapter 4

This chapter has used the well-known decomposition techniques reviewed in Chapter 2 to design utility-maximizing mechanisms for two networking technologies,

namely frequency-division multiple access and spatial reuse time-division multiple access.

In addition, we have presented a flow chart that can be useful in order to categorize and visualize previous results in the NUM literature. The flowchart highlights the inherent steps in posing, decomposing, and solving NUM optimization problems. Furthermore, the flowchart, with examples and a tutorial, is available on-line on the Internet. This opens up for the possibility of refining the flowchart and adding relevant results when they appear.

We have developed two center-free algorithms that optimally allocate the spectrum in a FDMA network. The first algorithm is based in dual relaxation and the second algorithm is based in primal decomposition. The algorithms have different time-scale properties.

Next, we have developed two algorithms to be used to devise a transmission schedule for a STDMA network. The algorithms are based on primal decomposition and on the premise that the schedule is augmented indefinitely. We have shown that the algorithms asymptotically reach the optimum under certain assumptions. However, an infinite schedule is not practical, and in practice, the schedule length is fixed. The most natural way of addressing this potential problem is to remove the oldest slot and replace it with a new slot. It turns out that this heuristic works very well in simulations.

7.2.1 Future Work

We need simple tuning rules that are robust. Most, if not all, results in the literature seem to depend rather critically on the network topology, and as is it now, engineers implementing these types of algorithms have to rely on trial and error in simulations and experiments to tune the algorithms.

Implementation experiments would be very interesting to perform; this is already taking place to some extent; see, e.g., Balucanti et al. (2009) and Warrier et al. (2008).

The heuristic to use a fixed number of slots in STDMA where the oldest slot is replaced with a new one works very well in simulations. However, the performance loss should be quantified and also bounded if possible.

There some recent results that allow us to re-write some nonconvex optimization problems into equivalent convex optimization problems. This opens up for interesting algorithms that can solve hard (at least in their nonconvex formulations) optimization problems without approximations; see, e.g., Papandriopoulos et al. (2006).

If TCP is used for congestion control, it would be interesting to study the dynamics of the resource allocation protocols using more accurate models, such as the novel models presented in Möller (2008) and Jacobsson (2008).

7.3 Chapter 5

Primal decomposition and incremental subgradient methods provide an interesting framework to pose distributed consensus problems. It has allowed us to consider general linear models for the agents and easily handle convex input constraints and linear state constraints. The convergence is mathematically guaranteed in the simplest case when negotiation and motion phases are separated.

Motivated by the many different algorithms that have been proposed for distributed averaging, we have investigated the convergence of a more general class of linear averaging iterations. First, we have considered linear iterations that allow nodes to maintain several internal states. We have provided necessary and sufficient conditions for convergence, and discussed some possible optimization schemes to improve the convergence rate.

It is known that a classical result can be used to compute the optimal weights for the single shift-register case; this provides us with a computationally inexpensive way to find a weight matrix A that converges faster (in the asymptotic sense) than the original algorithm in Xiao and Boyd (2004). A shift-register algorithm is better if high accuracy is desired, while the optimal symmetric algorithm has better initial convergence speed. In addition, numerical examples indicate that performance can be vastly improved if we use a nonsymmetric matrix A. However, this optimization problem is computationally intractable for larger network topologies.

For a truly distributed system, it is probably not possible to run any optimization problem at all to find a good (or optimal) A matrix, especially since the complete network topology needs to be known. In this case, the only viable methods are the heuristic methods presented in Section 2.3, which require only local topology information.

Finally, we looked at necessary and sufficient conditions for the convergence to consensus of a larger class of linear iterations with an augmented state vector. These conditions are not amenable for optimization in their present form.

7.3.1 Future Work

It is possible to use another optimization method instead of the incremental subgradient method in the distributed negotiation scheme. This could potentially lead to faster convergence and/or guaranteed progress per iteration.

A natural next step is to devise a convergence proof for an interleaved scheme, where interrupted negotiations are interleaved with application of the resulting control action in a receding horizon fashion. There have been some work recently done in this direction, namely: a study on general conditions for convergence of such schemes in Keviczky and Johansson (2008), and a convergence proof for a related problem with very special agent dynamics (integrators) in Ferrari-Trecate et al. (2007).

The consensus problem has received a significant amount of attention and the area is rather well understood. However, it would be interesting to see if the novel

general convergence conditions can be used to devise faster linear iterations.

7.4 Chapter 6

In this chapter, we have considered peer-to-peer algorithms for distributed optimization in wireless sensor networks. We have evaluated the performance of three specific algorithms on network nodes equipped with 802.15.4 compliant radios. While the linear iteration algorithm is tailored for averaging (least-squares) problems, where it excels in networks with few nodes, it is not applicable to general optimization problems. The other two algorithms are complementary and work surprisingly well in our simulations.

The simulations indicate that broadcast algorithms, where each node needs data from all of its neighbors to complete one iteration, do not scale with the number of nodes.

An interesting observation is that it is a design choice to put the stochastic behavior of the resulting system (optimization algorithm plus communication mechanism) either in the algorithm or in the communication mechanism. If we use a stochastic algorithm, in which the updates are explicitly stochastic, then we can use an unreliable communication mechanism with known time delay and number of required packets. On the other hand, if we use a deterministic algorithm, then we need a reliable communication mechanism, in which the time delay and number of packets needed are uncertain.

The algorithms require the graph to be symmetric, which turned out to be a bit problematic to guarantee with the simplistic neighbor discovery algorithm we used. We did not have this problem in the NS2 simulations, since we specified the radio ranges exactly to fit the generated graph. The simulations show that the algorithms work well also in a real WSN. However, since most of the algorithms are quite stochastic in their nature, more experiments need to be conducted in order for us to say something with more accuracy. The MISM have rapid initial convergence, but it will stay quite high all the way and it is fluctuating quite a bit.

Finally, our initial intuition that simple methods are very much needed have indeed been reinforced by our implementation experiences.

7.4.1 Future Work

The assumption that $(i, j) \in \mathcal{E}$ implies that system i can communicate with j is central in many of our problem formulations. Is this the correct abstraction? Our experiments indicate that the answer is in the negative, since motes that are far enough from each other to hinder reliable communications, may still be close enough to interfere with each other.

All of the implemented algorithms contain several parameters not present in the mathematical development of the optimization algorithms. Still, these parameters are crucial to the actual performance. How to tune them is a hard and open problem.

The algorithms we have used need to have a symmetric topology, which turned out to be problematic during our experiments. There are basically two solutions: either devise better (but still simple) neighbor discovery protocols that make symmetric topologies highly probable or change the algorithms to handle nonsymmetric topologies (directed graphs).

Appendix A

Notation

A.1 Symbols

Vectors are denoted with Latin characters, e.g., x, and are column vectors by default. There will be some exceptions, such as i, j, k, l, m, n, which are scalars and are mostly used for indexing. Otherwise, scalars are denoted with Greek characters, e.g., ε. Also here we have an exception, namely λ, which is sometimes a row vector. Matrices are denoted with capital letters, e.g., A. Functions will most often be denoted with Latin characters, e.g., f and g. Sets are denoted with calligraphic letters, e.g., $\mathcal{X}$. The use of symbols in this thesis follows the table below.

$\triangleq$	Definition
$\mathbb{N}$	Set of all natural numbers
$\mathbb{R}$	Set of all real numbers
$\mathbb{R}^{\xi}$	Set of all real vectors with ξ components
$\mathbb{R}^{\xi \times \zeta}$	Set of all real $\xi \times \zeta$ matrices
$\mathcal{G}$	The (undirected) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
$\mathcal{E}$	Set of edges (or links)
$\mathcal{V}$	Set of vertices (or nodes or agents)
$[x]_i$	The ith element of the vector x
$[A]_{ij}$	The element on row i and column j in the matrix A
A^{T}	Transpose of the matrix A
$^{\perp}$	Orthogonal complement
$\mathcal{R}(A)$	Range of matrix A
$\mathcal{N}(A)$	Nullspace of matrix A
$A^{\dagger}$	The Moore-Penrose pseudoinverse of the matrix A
A^{d}	The Drazin pseudoinverse of the matrix A

$\mathbf{1}_n$	Column vector with all n elements equal to one
$\mathbf{0}_n$	Column vector with all n elements equal to zero
I	Identity matrix
I_κ	Identity matrix in $\mathbb{R}^{\kappa\times\kappa}$
$\otimes$	Kronecker matrix product
$\nabla f(x)$	Gradient of f at x
$\partial f(x)$	The subdifferential, the set of all subgradients of f at x
$x > y$	$[x]_i > [y]_i$ for all i
$x \geq y$	$[x]_i \geq [y]_i$ for all i
$A > B$	The square matrix $A - B$ is positive definite
$A \geq B$	The square matrix $A - B$ is positive semi-definite
$\bar{x}$	Average of the vector x, $\bar{x} \triangleq \sum_i^n [x]_i/n$
$x^\star$	An optimizer of the optimization problem at hand
$f^\star$	The optimal objective value of the optimization problem at hand
$\mathcal{X}^\star$	The set of optimizers of the optimization problem at hand, $\mathcal{X}^\star \triangleq \{x \in \mathcal{X} \vert f(x) = x^\star\}$
$\mathcal{X}_\varepsilon$	The set of ε optimizers, $\mathcal{X}_\varepsilon \triangleq \{x \in \mathcal{X} \mid f(x) \leq f^\star + \varepsilon\}$
$\mathcal{B}_\eta$	Ball with radius η, $\mathcal{B}_\eta \triangleq \{x \vert \|x\| \leq \eta\}$
$x^{(k)}$	Iteration k of the vector x
L	Lagrangian
d	The dual function
$\mathbb{E}[x]$	Expected value of the random variable x
$\mathbb{P}\{\cdot\}$	Probability of the event $\{\cdot\}$
$I\{\cdot\}$	Indicator function for the event $\{\cdot\}$
$\mathcal{P}_\mathcal{X}[x]$	Euclidean projection of x on the closed convex set $\mathcal{X}$, $\mathcal{P}_\mathcal{X}[x] \triangleq \arg\min_z \{\|x - z\| \mid z \in \mathcal{X}\}$
$\mathrm{dist}_\mathcal{X}[x]$	Minimum distance from the point x to the set $\mathcal{X}$, $\mathrm{dist}_\mathcal{X}[x] \triangleq \inf\{\|x - z\| \mid z \in \mathcal{X}\}$
$\vert\mathcal{X}\vert$	The number of elements in the set $\mathcal{X}$
$\vert\varepsilon\vert$	Absolute value of ε
$\|x\|$	Euclidean norm, $\|x\| \triangleq \sqrt{x^\intercal x}$
$\|x\|_Q$	Weighted Euclidean norm, $\|x\|_Q \triangleq \sqrt{(Qx)^\intercal Qx}$, $Q > 0$
$\|A\|$	Spectral norm, $\|A\| \triangleq \sup_{x\neq 0} \|Ax\| \,/\, \|x\|$
$\|A\|_Q$	Weighted spectral norm, $\|A\|_Q \triangleq \sup_{x\neq 0} \|Ax\|_Q \,/\, \|x\|_Q$, $Q > 0$
$\rho(A)$	Spectral radius, $\rho(A) \triangleq \max_\lambda \{\vert\lambda\vert \mid Ax = \lambda x\}$

$\mathcal{W}$ $\quad \mathcal{W} \triangleq \{W \in \mathbb{R}^{n\times n} | [W]_{ij} = 0, (i,j) \notin \mathcal{E} \text{ and } i \neq j\}$

A.2 Acronyms

ACK	ACKnowledgment packet
AQM	Active Queue Management
BMI	Bilinear Matrix Inequality
CSMA-CA	Carrier Sense Multiple Access - Collision Avoidance
CSM	Consensus Subgradient Algorithm
DISM	Deterministic Incremental Subgradient Method
FDMA	Frequency-Division Multiple Access
HMC	Homogeneous Markov Chain
IEEE	Institute of Electrical and Electronics Engineers
KKT	Karush-Kuhn-Tucker
LMI	Linear Matrix Inequality
LQI	Link Quality Indicator
MAC	Medium Access Control
MISM	Markov Incremental Subgradient Method
MC	Markov Chain
NCRM	Non-smooth Center-free Resource allocation Method
NUM	Network Utility Maximization
OSI	Open Systems Interconnection
RISM	Randomized Incremental Subgradient Method
RSSI	Received Signal Strength Indicator
SDP	Semi-Definite Program
SINR	Signal to Interference Noise Ratio
STDMA	Spatial reuse Time-Division Multiple Access
TCP/IP	Transport Control Protocol / Internet Protocol
TDMA	Time-Division Multiple Access
WSN	Wireless Sensor Network

Appendix B

Numerical Experiments

B.1 Numerical Experiment Details for Chapter 3

The following probability transition matrix was used

$$P = \begin{pmatrix} 0.44951 & 0.2632 & 0 & 0 & 0.28729 \\ 0.2632 & 6.5155 \cdot 10^{-11} & 0.4736 & 0 & 0.2632 \\ 0 & 0.4736 & 1.9316 \cdot 10^{-10} & 0.5264 & 0 \\ 0 & 0 & 0.5264 & 0.4736 & 0 \\ 0.28729 & 0.2632 & 0 & 0 & 0.44951 \end{pmatrix}. \tag{B.1}$$

We have that $\rho(P - \mathbf{1}_n \mathbf{1}_n^\intercal / n) = 0.7846$.

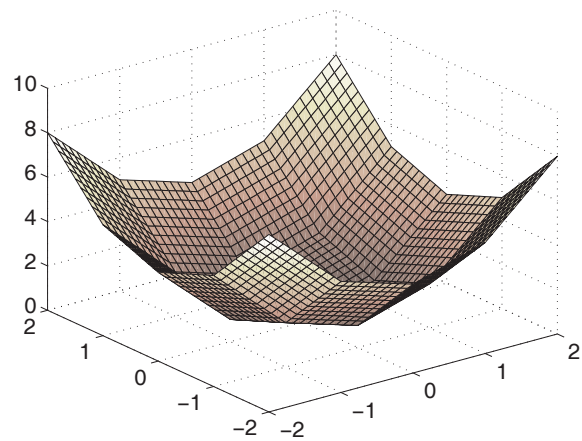


Figure B.1: Example of $\tilde{f}_i(x)$ with the parameters given in Table B.1.

Table B.1: Parameters for the example function in Fig. B.1.

Param.	Value	Param.	Value	Param.	Value
Q_1	$\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$	$\hat{x}_i$	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	β_1	1

Table B.2: Parameters for the objective function; see Fig. 3.4.

Param.	Value	Param.	Value	Param.	Value
Q_1	$\begin{pmatrix} 1.4 & 0 \\ 0 & 1 \end{pmatrix}$	$\hat{x}_1$	$\begin{pmatrix} -2.1 \\ -2.5 \end{pmatrix}$	β_1	1
Q_2	$\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$	$\hat{x}_2$	$\begin{pmatrix} 3.5 \\ -2.1 \end{pmatrix}$	β_2	1.5
Q_3	$\begin{pmatrix} .1 & 0 \\ 0 & .2 \end{pmatrix}$	$\hat{x}_3$	$\begin{pmatrix} 1.1 \\ -5.1 \end{pmatrix}$	β_3	1.5
Q_4	$\begin{pmatrix} 2.5 & 0 \\ 0 & 2 \end{pmatrix}$	$\hat{x}_4$	$\begin{pmatrix} -3.5 \\ 2.1 \end{pmatrix}$	β_4	.5
Q_5	$\begin{pmatrix} 1 & 0 \\ 0 & 1.5 \end{pmatrix}$	$\hat{x}_5$	$\begin{pmatrix} 2.1 \\ 3.1 \end{pmatrix}$	β_5	.75

The following consensus matrix was used

$$W = \begin{pmatrix} 0.33331 & 0.33333 & 0 & 0 & 0.33336 \\ 0.33333 & -0.16666 & 0.5 & 0 & 0.33333 \\ 0 & 0.5 & -1.4988 \cdot 10^{-6} & 0.5 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 \\ 0.33336 & 0.33333 & 0 & 0 & 0.33331 \end{pmatrix}. \tag{B.2}$$

We have that $\rho(W - \mathbf{1}_n\mathbf{1}_n^\intercal/n) = 0.7638$. Thus, it follows that $\rho(P - \mathbf{1}_n\mathbf{1}_n^\intercal/n) > \rho(W - \mathbf{1}_n\mathbf{1}_n^\intercal/n)$, as expected, since the problems of finding W and P are almost identical, except that the feasible set is smaller for P (non-negativity of the elements).

Bibliography

L. Ahlin and J. Zander. *Principles of Wireless Communications*. Studentlitteratur (1998).

I. F. Akyildiz, S. Weilian, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8): 102–114 (2002).

E. Alba and J. M. Troya. A survey of parallel distributed genetic algorithms. *Complexity*, 4(4): 31–52 (1999).

C. Antal, J. Molnár, S. Molnár, and G. Szabó. Performance study of distributed channel allocation techniques for a fast circuit switched network. *Computer Communications*, 21(17): 1597–1609 (1998).

K. J. Arrow and L. Hurwicz. Decentralization and computation in resource allocation. In P. W. Pfouts, editor, *Essays in Economics and Econometrics*, 34–104. Chapel Hill: University of North Carolina Press (1960).

K. J. Arrow, L. Hurwicz, and H. Uzawa. *Studies in Linear and Non-linear Programming*. Stanford University Press (1958).

L. Balucanti, M. Belleschi, P. Soldati, M. Johansson, and A. Abrardo. An optimal cross-layer protocol for ds-cdma ad-hoc networks: design and implementation. In *IEEE Infocom* (2009). Submitted.

N. Bambos, S. C. Chen, and G. J. Pottie. Channel access algorithms with active link protection for wireless communication networks with power control. *IEEE/ACM Transactions on Networking*, 8(5): 583–597 (2000).

S. Barbarossa and G. Scutari. Decentralized maximum likelihood estimation for sensor networks composed of nonlinearly coupled dynamical systems. *IEEE Transactions on Signal Processing*, 55: 3456–3470 (2007).

A. Ben-Israel and T. N. E. Greville. *Generalized inverses: theory and appliations*. John Wiley & Sons (1974).

D. P. Bertsekas. *Nonlinear programming*. Athena Scientific (1999).

D. P. Bertsekas, A. Nedić, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific (2003).

D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific (1997).

D. Blatt, A. Hero, and H. Gauchman. A convergent incremental gradient method with a constant step size. *SIAM J. Optim.*, 18(1): 29–51 (2007).

S. Boyd, P. Diaconis, and L. Xiao. Fastest mixing markov chain on a graph. *SIAM Review*, 46: 667–689 (2004).

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press (2004).

P. Brémaud. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. Springer-Verlag (1999).

R. Bush and D. Meyer. Some Internet Architectural Guidelines and Philosophy. RFC 3439 (Informational) (2002). URL `http://www.ietf.org/rfc/rfc3439.txt`.

M. Cao, D. A. Spielman, and E. M. Yeh. Accelerated gossip algorithms for distributed computation. In *Allerton Conference* (2006).

C. M. Carretti. *Comparison of Distributed Optimization Algorithms in Sensor Networks*. Master's thesis, Royal Institute of Technology (KTH) (2008).

M. Chiang. Balancing transport and physical layers in wireless multihop networks: jointly optimal congestion control and power control. *IEEE Journal on Selected Areas in Communications*, 23(1): 104–116 (2005).

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 (2007).

K. L. Chung. *A Course in Probability Theory*. Academics Press (1974).

J. Cortéz, S. Martínez, and F. Bullo. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Transactions on Automatic Control*, 51(8): 1289–1298 (2006).

D. Culler, D. Estrin, and M. Srivastava. Guest editors' introduction: Overview of sensor networks. *Computer*, 37(8): 41–49 (2004).

D. H. Cushing and F. R. H. Jones. Why do fish school? *Nature*, 218: 918–920 (1968).

G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7: 279–301 (1989).

J. M. Danskin. *Theory of Max-Min*. Springer-Verlag Berlin (1967).

M. H. DeGroot. Reaching a consensus. *Journal of the American Statistical Association*, 69(345): 118–121 (1974).

V. Dem'yanov and V. Shomesova. Subdifferentials on functions on sets. *Kibernetika*, (1): 23–27 (1980).

R. Diestel. *Graph Theory*. Springer-Verlag (2005).

W. B. Dunbar and R. M. Murray. Distributed receding horizon control with application to multi-vehicle formation stabilization. *Automatica*, 42(4): 549–558 (2006).

J. C. Dunn and S. Harshbarger. Conditional gradient algorithms with open loop step size rules. *Journal of Mathematical Analysis and Applications*, 62: 432–444 (1978).

P. Dutta and D. Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *Proceedings of ACM SenSys* (2008).

S. Elhedhli, J. Goffin, and J. Vial. Nondifferentiable optimization: Cutting plane methods. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, volume 4, 40–45. Kluwer Academic Publishers (2001).

Y. M. Ermol'ev. Methods of solution of nonlinear extremal problems. *Cybernetics*, 2(4): 1–17 (1966).

G. Ferrari-Trecate, L. Galbusera, M. Marciandi, and R. Scattolini. A model predictive control scheme for consensus in multi-agent systems with single-integrator dynamics and input constraints. In *IEEE CDC* (2007).

O. E. Flippo and A. H. G. Rinnooy Kan. Decomposition in general mathematical programming. *Mathematical Programming*, 60: 361–382 (1993).

G. Foschini and Z. Miljanic. A simple distributed autonomous power control algorithm and its convergence. *IEEE Transactions on Vehicular Technology*, 42(4): 641–646 (1993).

C. Godsil and G. Royle. *Algebraic Graph Theory*. Springer-Verlag (2001).

S. J. Golestaani. *A unified theory of flow control and routing in data communication networks*. Ph.D. thesis, MIT (1979).

G. H. Golub and R. S. Varga. Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order richardson iterative methods. *Numerische Matematik*, 3: 147–156 (1961).

B. Gosh, S. Muthukrishnan, and M. H. Schultz. First and second order diffusive methods for rapid, coarse, distributed load balancing. In *ACM symposium on Parallel algorithms and architectures* (1996).

J. Grönkvist. *Interference-Based Scheduling in Spatial Reuse TDMA*. Ph.D. thesis, Royal Insititute of Technology (KTH) (2005).

E. B. Hamida, G. Chelius, and E. Fleury. Neighbor discovery analysis in wireless sensor networks. In *Proceedings of ACM CoNEXT* (2006).

W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57: 97–109 (1970).

Y. C. Ho and D. L. Pepyne. Simple explanation of the no-free-lunch theorem and its implications. *Journal of Optimization Theory and Applications*, 115(3): 549–570 (2002).

Y. C. Ho, L. Servi, and R. Suri. A class of center-free resource allocation algorithms. *Large Scale Systems*, 1: 51–62 (1980).

K. Holmberg. Primal and dual decomposition as organizational design: price and/or resource directive decomposition. In R. M. Burton and B. Obel, editors, *Design models for hierarchical organizations: computation, information, and decentralization*. Kluwer Academic Publishers (1995).

K. Holmberg and K. Kiwiel. Mean value cross decomposition for nonlinear convex problems. *Optimization Methods and Software*, 21(3) (2006).

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press (1985).

T. Ibaraki and N. Katoh. *Resource Allocation Problems*. Foundations of computing. MIT Press (1988).

IEEE. IEEE Standard 802.15.4 (2006). URL `http://standards.ieee.org/`.

K. Jacobsson. *Dynamic modeling of Internet congestion control*. Ph.D. thesis, Royal Institute of Technology (KTH) (2008).

B. Johansson, C. Adam, M. Johansson, and R. Stadler. Distributed resource allocation strategies for achieving quality of service. In *Proceedings of IEEE CDC*, 1990–1995 (2006*a*).

B. Johansson, C. M. Carretti, and M. Johansson. On distributed optimization using peer-to-peer communications in wireless sensor networks. In *Proceedings of IEEE SECON* (2008*a*).

B. Johansson and M. Gäfvert. Untripped SUV rollover detection and prevention. In *Proceedings of IEEE CDC* (2004).

B. Johansson and M. Johansson. Primal and dual approaches to distributed cross-layer optimization. In *16th IFAC World Congress* (2005).

B. Johansson and M. Johansson. Faster linear iterations for distributed averaging. In *IFAC World Congress* (2008).

B. Johansson, T. Keviczky, K. H. Johansson, and M. Johansson. Subgradient methods and consensus algorithms for solving convex optimization problems. In *Proceedings of IEEE CDC* (2008*b*).

B. Johansson, H. Li, J. Huang, M. Chiang, and M. Johansson. Network utility maximization website (2008*c*). URL `http://www.networkutilitymaximization.org/`.

B. Johansson, M. Rabi, and M. Johansson. A simple peer-to-peer algorithm for distributed optimization in sensor networks. In *Proceedings of IEEE CDC* (2007).

B. Johansson, M. Rabi, and M. Johansson. A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM J. Optim.* (2008*d*). Submitted.

B. Johansson, P. Soldati, and M. Johansson. Mathematical decomposition techniques for distributed cross-layer optimization of data networks. *IEEE Journal on Selected Areas in Communications*, 24(8): 1535–1547 (2006*b*).

B. Johansson, A. Speranzon, M. Johansson, and K. H. Johansson. Distributed model predictive consensus. In *Mathematical Theory of Networks and Systems* (2006*c*).

B. Johansson, A. Speranzon, M. Johansson, and K. H. Johansson. On decentralized negotiation of optimal consensus. *Automatica*, 44: 1175–1179 (2008*e*).

M. Johansson and L. Xiao. Cross-layer optimization of wireless networks using nonlinear column generation. *Wireless Communications, IEEE Transactions on*, 5(2): 435–445 (2006).

V. Kawadia and P. R. Kumar. A cautionary perspective on cross-layer design. *IEEE Wireless Communications Magazine*, 12(1): 3–11 (2005).

F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49: 237–252 (1998).

J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Van Nostrand (1960).

T. Keviczky, F. Borrelli, and G. J. Balas. Decentralized receding horizon control for large scale dynamically decoupled systems. *Automatica*, 42(12): 2105–2115 (2006).

T. Keviczky and K. H. Johansson. A study on distributed model predictive consensus. In *IFAC World Congress* (2008).

V. M. Kibardin. Decomposition into functions in the minimization problem. *Automation and Remote Control*, 40(9): 1311–1323 (1980).

K. C. Kiwiel. Convergence of approximate and incremental subgradient methods for convex optimization. *SIAM J. Optim.*, 14(3): 807–840 (2004).

M. Kocvara and M. Stingl. PENNON - a code for convex nonlinear and semidefinite programming. *Optimization Methods and Software*, 18: 317–333 (2003).

H. J. Kushner and G. G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer (2003).

J. Lagarias, J. Reeds, M. Wright, and P. Wright. Convergence properties of the nelder-mead simplex method in low dimensions. *SIAM Journal of Optimization*, 9: 112–147 (1998).

T. Larsson, M. Patriksson, and A.-B. Strömberg. Ergodic, primal convergence in dual subgradient schemes for convex programming. *Mathematical Programming*, 86: 283–312 (1999).

L. S. Lasdon. *Optimization Theory for Large Systems*. Macmillan (1970).

X. Lin and N. B. Shroff. The impact of imperfect scheduling on cross-layer rate control in multihop wireless networks. In *Proceedings of IEEE INFOCOM* (2005).

J. Löfberg. Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference* (2004).

Z. Lotker, B. PattShamir, and A. Rosén. Distributed approximate matching. In *ACM PODC* (2007).

S. H. Low. A duality model of TCP and Queue Management Algorithms. *IEEE/ACM Trans. on Networking*, 4(11): 525–536 (2003).

S. H. Low and D. E. Lapsley. Optimization flow control – I: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6): 861–874 (1999).

D. G. Luenberger. *Optimization by Vector Space Methods*. John Wiley & Sons (1969).

N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers (1996).

J. M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall (2002).

B. Maciel. *Development of a Detection System using a Wireless Sensor Network*. Master's thesis, Royal Institute of Technology (KTH) (2008).

T. A. Manteuffel. The tchebychev iteration for nonsymmetric linear systems. *Numerische Matematik*, 28: 307–317 (1977).

T. A. Manteuffel. Optimal parameters for linear second-degree stationary methods. *SIAM Journal on Numerical Analysis*, 19(4): 833–839 (1982).

J. R. Marden, G. Arslan, and J. S. Shamma. Connections between cooperative control and potential games illustrated on the consensus problem. In *European Control Conference* (2007).

J. Markoff. Can't Find a Parking Spot? Check Smartphone. *The New York Times* (2008). URL `http://www.nytimes.com/`.

N. D. Mermin. What's wrong with this lagrangean? *Physics Today*, 41(4): 9–11 (1988).

C. D. Meyer and R. J. Plemmons. Convergent powers of a matrix with applications to iterative methods for singular linear systems. *SIAM Journal on Numerical Analysis*, 14: 699–705 (1977).

N. Möller. *Window-based congestion control—Modeling, analysis and design*. Ph.D. thesis, Royal Institute of Technology (KTH) (2008).

A. Nedić. *Subgradient Methods for Convex Minimization*. Ph.D. thesis, MIT (2002).

A. Nedić and D. P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM J. Optim.*, 12(1): 109–138 (2001).

A. Nedić and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. Technical report, MIT (2007*a*). LIDS technical report 2755.

A. Nedić and A. Ozdaglar. On the rate of convergence of distributed asynchronous subgradient methods for multi-agent optimization. In *Proceedings of IEEE CDC* (2007*b*).

A. S. Nemirovski and D. B. Judin. Cesari convergence of the gradient method of approximating saddle points of convex-concave functions. *Soviet Math. Dokl.*, 19(2): 482–486 (1978).

J. R. Norris. *Markov Chains*. Cambridge University Press (1998).

NS2. Network simulator website (2008). URL `http://www.isi.edu/nsnam/ns/`.

B. Obel. A note on mixed procedures for decomposing linear programming problems. *Mathematische Operationsforschung und Statistik, Series Optimization*, 9: 537–544 (1978).

R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1): 215–233 (2007).

A. Olshevsky and J. N. Tsitsiklis. Convergence rates in distributed consensus and averaging. In *Proceedings of IEEE CDC* (2006).

M. Padlipsky. Perspective on the ARPANET reference model. RFC 871 (1982). URL `http://www.ietf.org/rfc/rfc871.txt`.

D. P. Palomar and M. Chiang. A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications*, 24(8): 1439–1451 (2006).

J. Papandriopoulos, S. Dey, and J. S. Evans. Optimal and distributed protocols for cross-layer design of physical & transport layers in MANETs. *IEEE/ACM Transactions on Networking* (2006). Submitted.

M. Patriksson. A survey on the continuous nonlinear resource allocation problem. *European Journal of Operational Research*, 185(1): 1–46 (2008).

J. G. Proakis. *Digital Communications*. McGraw-Hill (2001).

QCN. Quake-Catcher Network website (2008). URL `http://qcn.stanford.edu/`.

M. Rabbat and R. Nowak. Distributed optimization in sensor networks. In *Proceedings of ACM/IEEE IPSN* (2004).

M. Rabbat, R. Nowak, and J. Bucklew. Generalized consensus computation in networked systems with erasure links. In *Proceeding of IEEE SPAWC* (2005).

R. L. Raffard, C. J. Tomlin, and S. P. Boyd. Distributed optimization for cooperative agents: Applications to formation flight. In *Proceedings of IEEE CDC* (2004).

S. S. Ram, A. Nedić, and V. Veeravalli. Incremental stochastic subgradient algorithms for convex optimzation (2008). Submitted.

R. T. Rockafellar. *Convex Analysis*. Princeton Univeristy Press (1970).

R. Rockafellar. *Network flows and monotropic optimization*. John Wiley & Sons (1984).

R. Rom and M. Sidi. *Multiple Access Protocols: Performance and Analysis*. Springer-Verlag (1990).

W. Rudin. *Principles of Mathematical Analysis*. McGraw-Hill (1976).

I. Schizas, A. Ribeiro, and B. Giannakis. Consensus-based distributed parameter estimation in ad hoc wireless sensor networks with noisy links. In *Proceedings of IEEE ICASSP* (2007).

SeDuMi. SeDuMi website (2008). URL `http://sedumi.mcmaster.ca/`.

N. Z. Shor. *On the structure of algorithms for the numerical solution of optimal planning and design problems*. Ph.D. thesis, Cybernetics Institute (1964).

N. Z. Shor. *Minimization methods for non-differentiable functions*. Springer-Verlag (1985).

S. Skogestad and I. Postlethwaite. *Multivariable Feedback Control*. John Wiley & Sons, New York-Chichester-Brisbane (2005).

P. Soldati, B. Johansson, and M. Johansson. Distributed optimization of end-to-end rates and radio resources in wimax single-carrier networks. In *Proceedings of IEEE GLOBECOM* (2006*a*).

P. Soldati, B. Johansson, and M. Johansson. Proportionally fair allocation of end-to-end bandwidth in STDMA wireless networks. In *Proceedings of ACM MobiHoc* (2006*b*).

P. Soldati, B. Johansson, and M. Johansson. Distributed cross-layer coordination of congestion control and resource allocation in s-TDMA wireless networks. *Wireless Networks*, 14: 949–965 (2008).

S.-H. Son, M. Chiang, S. R. Kulkarni, and S. C. Schwartz. The value of clustering in distributed estimation for sensor networks. In *IEEE Wirelesscom* (2005).

A. Speranzon, C. Fischione, B. Johansson, and K. H. Johansson. Adaptive distributed estimation over wireless sensor networks with packet losses. In *Proceedings of IEEE CDC* (2007).

R. Srikant. *The Mathematics of Internet Congestion Control*. Birkhäuser (2004).

K. Srinivasan and P. Levis. Rssi is under appreciated. In *Proceedings of EmNetS* (2006).

TinyOS. Tinyos website (2008). URL `http://www.tinyos.net/`.

Tmote. Tmote sky datasheet (2008). URL `http://www.sentilla.com/pdf/eol/tmote-sky-datasheet.pdf`.

J. N. Tsitsiklis. *Problems in decentralized decision making and computation*. Ph.D. thesis, MIT (1984).

J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9): 803–812 (1986).

R. H. Tutuncu, K. C. Toh, and M. J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical Programming Ser. B*, 95: 189–217 (2003).

J. van Ast, R. Babůska, and B. D. Schutter. Particle swarms in optimization and control. In *IFAC World Congress* (2008).

T. J. Van Roy. Cross decomposition for mixed integer linear programming. *Mathematical Programming*, 25: 46–63 (1983).

R. Varga. *Matrix Iterative Analysis*. Prentice-Hall (1962).

B. Wang. *Distributed Resource Allocation and Performance Optimization for Video Communication over Mesh Networks Based on Swarm Intelligence*. Ph.D. thesis, University of Missouri-Columbia (2007).

J. Wang, L. Li, S. H. Low, and J. C. Doyle. Cross-layer optimization in tcp/ip networks. *IEEE/ACM Trans. on Networking*, 3(13): 582 – 595 (2005).

A. Warrier, S. Ha, P. Wason, I. Rhee, and J. H. Kim. Diffq: Differential backlog congestion control for wireless multi-hop networks. In *IEEE SECON* (2008).

D. X. Wei, C. Jin, S. H. Low, and S. Hegde. Fast tcp: Motivation, architecture, algorithms, performance. *Networking, IEEE/ACM Transactions on*, 14(6): 1246–1259 (2006).

L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1): 65–78 (2004).

L. Xiao and S. Boyd. Optimal scaling of a gradient method for distributed resource allocation. *J. Opt. Theory and Applications*, 129(3): 469–488 (2006). Submitted.

L. Xiao, M. Johansson, and S. Boyd. Simultaneous routing and resource allocation via dual decomposition. *IEEE Transactions on Communications*, 52(7): 1136–1144 (2004).

D. M. Young. Second-degree iterative methods for the solution of large linear systems. *Journal of Approximation Theory*, 5: 137–148 (1972).

J. Zheng and M. Lee. A comprehensive performance study of IEEE 802.15.4. *Sensor Network Operations, IEEE press*, Wiley Interscience, chapter 4: 218–237 (2006).

H. Zimmermann. OSI reference model - the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 4: 425–432 (1980).

"*Författarnas lättnad överträffas nog bara av läsarens när vi nu förklarar denna lärobok i matematik för avslutad.*"

A. Persson and L.-C. Böiers, Analys i en variabel, 1990.

TRITA-EE 2008:065
ISSN 1653-5146
ISBN 978-91-7415-190-9