Scheduling in Cyber-Physical Systems

Submitted in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the

Electrical and Computer Engineering

Dissertation by

Qiao Li

M.S., Electrical and Computer Engineering, Carnegie Mellon University

B.E., Electronics Information Engineering, Tsinghua University

Carnegie Institute of Technology

Carnegie Mellon University

Pittsburgh, Pennsylvania

August, 2012

Scheduling in Cyber-Physical Systems

ABSTRACT

Cyber-physical systems (CPS) refer to a promising class of systems featuring intimate coupling between the 'cyber' intelligence and the 'physical' world. Enabled by the ubiquitous availability of computation and communication capabilities, such systems are widely envisioned to redefine the way that people interact with the physical world, similar to the revolutionary role of internet in transforming how people interact with each other. As the whole society becomes increasingly dependent on such systems, it is crucial to develop a theory to understand and optimize the CPS in a systematic manner.

This thesis contributes to the foundations of CPS by identifying and addressing a general class of scheduling-type applications for a vital class of CPS, the physical networks (PhyNets). Different from the abstract CPS, a PhyNet has a graph-type physical part, which represents the local interactions among users in the system, as specified by certain well-known physical laws. Thus, it is very promising to develop efficient distributed algorithms in PhyNets with proper communication infrastructure and protocols, due to the physical graph structure. The 'scheduling' refers to the applications where joint actions of all users are coordinated, in order to allocate system resources to satisfy certain long term and uncertain demands. Important applications of the scheduling in PhyNets include packet scheduling in wireless networks, coordinated charging of electric vehicles (EV) in electric power grids, and workload scheduling in data centers. In this thesis, we assume very mild assumptions on the stochastic processes, and provide probabilistic scheduling performance guarantees using the technique of fluid limits.

In this thesis, we will investigate a broad range of scheduling algorithms and discuss their performance and distributed implementation. We first investigate the class of optimal scheduling algorithms in the dynamic regime, where the system modes change randomly with time. We focus on augmented max-weight scheduling schemes, which choose a max-weight schedule, where the

weight is specified by queue lengths. Two scenarios are considered in this case. For the first scenario, we assume the scheduler has asymptotic knowledge about the optimal cost, and propose virtual cost queue based max-weight scheduling schemes. We prove cost optimality and rate stability results using fluid limits. For the second scenario, we assume no knowledge on optimal cost, and adopt a Lyapunov optimization based approach. We demonstrate the asymptotic optimality and provide bounds on the average queue lengths. Finally, we apply the augmented max-weight algorithms to the important application of coordinated EV charging in power systems.

We next consider the class of optimal scheduling algorithms in the quasi-static regime, where the system modes remain constant for the scheduling application. The quasi-static property is promising for efficient scheduling design by allowing the system to 'memorize' good schedules. We propose a simplex algorithm based scheduling scheme, and prove that it is asymptotically throughput optimal. For the important application of packet scheduling in wireless networks, we show that the simplex scheduling can be implemented in a distributed manner with average consensus and carrier sensing multiple access (CSMA) mechanisms. We also demonstrate that it achieves significant steady-state delay reduction compared to the popular throughput optimal distributed adaptive CSMA schemes, by successfully avoiding the random walk behavior associated with the distributed CSMA.

Finally, we investigate the performance of suboptimal scheduling schemes. We will discuss the performance of a class of interesting scheduling schemes, maximal scheduling. A maximal scheduling algorithm only involves simple and local coordination among users, and therefore has low complexity and is easy for distributed implementation. We propose a tight lower bound throughput region for maximal scheduling algorithms, and show that it can achieve a certain fraction of the optimal region. We also investigate the performance improvement on maximal scheduling. In particular, for packet scheduling in wireless networks, we propose a static priority assisted maximal scheduling scheme. We show that the optimal static priority assignment can be computed with low complexity in an online manner, and that the combined priority assignment and maximal scheduling achieve dramatic throughput improvement over the conventional maximal scheduling.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Professor Rohit Negi, for his support, inspiration, encouragement and trust, without which this thesis can never be possible. It was my fortune and privilege to work with him. Professor Negi has always been a fountain of knowledge for me no matter what issue challenged me during the course of my work. I am truly grateful for the research freedom he gave me on developing expertise of my own, his willingness to share his deep knowledge and strong analytical skills whenever I was at stuck, and his encouragement and support on my research and choice of career path.

I am very grateful for my outstanding thesis committee members – Professor José Moura, Professor Marija Ilić and Professor R. Srikant. Professor Moura has been a wonderful source of insight and enthusiasm that has helped me identify the strengths and improve the weaknesses of my thesis work. I would miss the enjoyable individual discussions with Professor Ilić, and would like to thank her for the valuable advice and support on my job applications. I have learned immensely from Professor Srikant's research during the past years, and truly appreciate his deep and insightful comments, which have greatly improved this thesis.

I would like to thank my colleagues and friends at Carnegie Mellon. Thanks to Gyouhwan Kim and Satashu Goel, who have always been willing to offer help during my first year; to Balakrishnan Narayanaswamy, for the stimulating discussions even years after his graduation; to Andrew Cheng, Sungchul Han, Euiseok Hwang, Vinay Prabhu, Yaron Rachlin, Arjunan Rajeswaran, and Yang Weng, thanks to all of you for the warm and friendly lab atmosphere and the sharing of your knowledge during the group discussions. Special thanks to Andrew Cheng for carefully proofreading the manuscript. I would also like to thank the group of friends and colleagues in the ECE department. Special thanks to the folks in Electric Energy Systems Group (EESG), in particular Tao Cui, Qixing Liu, Rui Yang and Dinghua Zhu, for discussing smart grid research projects, and

answering my questions about power systems. Thanks to Vishnu Naresh Boddeti, Yu Cai, Pablo Hennings, Xinde Hu, Seungjune Jeon, Soowoong Lee, Congcong Li, Sheida Nabavi, Yibin Ng, Xiaohui Wang, Can Ye and Wei Yu, among many others, for making my study in Carnegie Mellon such an enjoyable and rewarding process.

This work would not have been possible without the support from my family. I am indebted to the support and encouragement from my parents. My Mom and Dad have always cared for me, tried their best to understand every detail of my work and provide suggestions. I am blessed to have the companionship of my wife, Jingyuan Huang, for going through the ups and downs with me throughout these years at Carnegie Mellon. It is my fortune to always have you by my side.

Finally, this thesis work was supported by the U.S. National Science Foundation under awards CNS-0831973 and ECCS-0931978, and by U.S. Army Research Office under award W911NF0710287. I greatly appreciate the financial support.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The rapid development of information technologies in the past decades has resulted in wide availability of embedded computing and communication capabilities in almost all types of objects. Such large-scale and deep embedding of the cyber intelligence into the physical world has created unprecedented opportunities for researchers to develop systems with huge societal impacts and economic benefits. Commonly referred to as the cyber-physical systems (CPS) [1–3], these systems are envisioned to achieve important functionalities that cannot be achieved previously, by utilizing the intimate coupling of the 'cyber' core with the 'physical' environment. The CPS is an emerging and hot research area, covering a broad range of sectors, with important applications ranging from macro-scale infrastructure based systems, such as smart grid [4], data centers [5, 6], transportation systems [7], to micro-scale systems, such as intelligent medical devices [8]. It is widely envisioned that the CPS will play such an important role that it will redefine the way people interact with the physical world, similar to the way internet revolutionized the way that people interact with each other.

The CPS research is both very important and highly challenging, which covers a diverse range of areas. Thus, it is important to develop theoretical foundations to understand and design such systems in a systematic manner. Realizing this important goal, in this thesis we contribute to the foundations of CPS by addressing a class of important applications, all of which share a common structure, so

that similar techniques can be brought to bear in each case. Specifically, this thesis focuses on the *scheduling* applications for a vital class of CPS, the *physical networks* (PhyNets). The 'scheduling' refers to applications where certain resources in the system are allocated by coordinating all users to satisfy uncertain and long-term average demands. One important example is packet scheduling in wireless networks, where the scarce wireless spectrum has to be allocated across all links in the network, to satisfy each link's traffic demand. We are interested in investigating such scheduling-type problems in the context of PhyNets, where the graph structure of the physical plant allows efficient and distributed implementations. For the remaining of this chapter, we will provide a brief introduction to the general scheduling problem, state our contributions and provide a summary of related work. We first introduce the model of PhyNets and discuss the scheduling with PhyNets.

1.1 CYBER-PHYSICAL SYSTEMS AND PHYSICAL NETWORKS

Cyber-physical systems are advanced engineering systems where the computing and communication are carefully designed to achieve intimate integration with the physical dynamics. An example structure of the general CPS is illustrated in Fig. 1.1. The typical CPS has three major parts. The first part is the *physical plant*, which is an abstraction of the physical world. The second part consists of many *platforms*, which are equipped with sensors, computing devices and actuators. Finally, these platforms are interconnected by the third part, namely a *communication network*, so that the operations of all platforms can be coordinated to achieve desired functionalities with the physical plant. The platforms and the communication network form the 'cyber part' of the CPS, whereas the physical plant represents the 'physical part' of the CPS.

The abstract structure of CPS, as shown in Fig. 1.1, is very general, which can be used to model an enormous class of systems, from national infrastructures such as the power grid to small cardiac medical devices. However, such level of abstraction in modeling makes it extremely challenging, if not impossible, for researchers to address the CPS design and analysis in a unified manner. A core issue is that the 'physical plant', as shown in Fig. 1.1, does not provide any insight into the problem

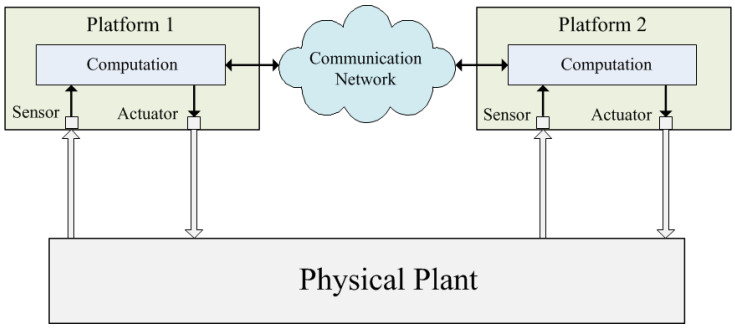


Figure 1.1: An example structure of a typical CPS.

structure in its full generality, and therefore is too abstract for efficient analysis and design. As an alternative, this thesis focuses on one specific class of CPS, *physical networks*, where the abstract physical plant can be modeled by a 'physical graph'. An example of the PhyNet is illustrated in Fig. 1.2. Compared to the architecture of general CPS in Fig. 1.1, the most important feature of a PhyNet is that its physical plant can be abstracted by a much simpler physical factor graph $\mathcal{G}$. For the physical graph $\mathcal{G}$, each variable node represents a *user* in the system, which corresponds to a concrete physical entity in the physical world, such as a link in wireless networks, and a server in data centers. The factor nodes represent network coupling among the users, due to certain well-known physical laws. It is somewhat surprising that a wide variety of physical laws can be described or approximated as local interactions, such as the conservation laws. Thus, the PhyNet model can potentially be used for many important CPS applications.

Compared to the abstract CPS structure, the physical graph representation in a PhyNet is promising to achieve efficient and distributed algorithms. In this thesis, we will propose a wide range scheduling algorithms and show that they all can be implemented in a distributed manner, using techniques such as dual decomposition, average consensus, and statistical sampling. The specific implementation method, on the other hand, should be based on the structure of the particular application. We emphasize that all such distributed implementation methods can be applied due to the

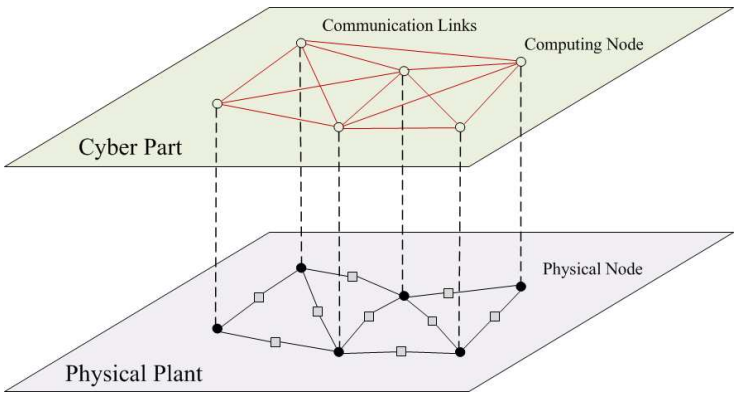


Figure 1.2: An example structure of a PhyNet.

critical assumption that the physical plant can be modeled as a graph.

1.2 SCHEDULING IN PHYNETS

This thesis considers one important type of applications in PhyNets, namely scheduling problems. The 'scheduling' in this thesis is a general definition, which refers to applications where resources in the system are efficiently allocated to satisfy certain long term and uncertain average demands. In below, we will briefly discuss the motivations and applications of the scheduling problem in the context of different CPS applications:

- *Packet Scheduling in Wireless Networks*

  As one important application of the scheduling framework, the packet scheduling in wireless networks has been subject to extensive studies in the past [9–26]. For such applications, the resource in the system corresponds to the scarce wireless spectrum, which has to be efficiently allocated among users in the network to satisfy their packet traffic demands. For such problems, the physical graph corresponds to the well-known interference graph [27], which specifies that two links which are connected by an edge (or equivalently, a factor node) cannot transmit together, due to the strong co-channel interference. We will discuss this model

in detail in Chapter 2, where we will also present a hypergraph interference model for the cumulative co-channel interference.

- *Coordinated Charging of Electric Vehicles in Power Systems*

  Another important application of the scheduling formulation is the coordinated charging of electric vehicles (EV) in power systems [28–37], which is an emerging and hot research topic in smart grids. It is widely envisioned that [30] [38] the current power system infrastructure can only support a small EV penetration level (such as 10%) if all EVs charge in an uncoordinated fashion, due to the severe congestion issues and voltage problems during peak load periods. Thus, for the EV charging problem, it is important to allocate the 'active power resource' in the system to all EV users efficiently, so as to satisfy their energy needs, while guaranteeing that the power system can operate in a secure and reliable manner. The physical graph for the EV charging application corresponds to the AC power flow coupling, which is a special case of the conservation law. We will discuss the detailed modeling in Chapter 2.

- *Workload Scheduling in Data Centers*

  Finally, we will show that the scheduling formulation can include workload scheduling in data centers [39–44] as a special case. We are particularly interested in thermal-aware workload scheduling applications. Thermal issues have been considered as a dominating problem for the efficient and reliable operation of data centers [40, 45], as they can affect both the performance of the processors and the cooling efficiency. Thus, it is desired to allocate the 'computing power resource' among all processors in the system efficiently, so as to satisfy the workload requirements for each processor, while maintaining desired temperature profiles for all processors. In this case, the physical graph models the thermal coupling among different processors, in that one processor's speed may affect the temperature of a 'local' subset of processors, due to the heat energy conservation law. We will discuss the modeling in detail in Chapter 2.

1.3 SUMMARY OF CONTRIBUTIONS

This thesis proposes a general scheduling framework for an important class of CPS. We will demonstrate that the framework can be used for a diverse range of applications, from packet scheduling in wireless networks, to EV charging in smart grids and workload scheduling in data centers. We will investigate both optimal scheduling schemes and suboptimal scheduling schemes, discuss their distributed implementations, and demonstrate their performance in the context of important applications. Here is a brief summary of the key contributions of this thesis, which are listed in a chapter-wise manner.

- Chapter 2 proposes the general scheduling problem with PhyNets, and shows that it includes many CPS applications. We will demonstrate three applications mentioned in the previous section in detail.

- Chapter 3 considers optimal scheduling schemes in the dynamic regime, where the system modes change randomly across different time slots. In the case with asymptotic knowledge about the optimal scheduling cost, we propose virtual queue based max-weight scheduling schemes and prove the optimality results using fluid limits. Two scheduling algorithms will be presented. The first one is a generalization of the conventional max-weight scheduling, whereas the second one is a generalized 'pick-and-compare' algorithm, which has low complexity and is easy to be implemented in a distributed manner, using average consensus techniques. In the second case without knowledge about the optimal scheduling cost, we will propose a Lyapunov optimization based max-weight policy and prove its asymptotic optimality with Lyapunov drift analysis. We will finally apply the max-weight scheduling schemes to the important application of coordinated EV charging in power systems.

- Chapter 4 addresses optimal scheduling in the quasi-static regime, where the system modes remain unchanged for the scheduling problem. We propose a simplex algorithm based schedul-

ing scheme, and prove its optimality using fluid limits. We show that the scheduling can be implemented in a distributed manner using average consensus techniques. The distributed scheduling incurs higher complexity than the 'pick-and-compare' scheduling in Chapter 3. On the other hand, the algorithm achieves significant delay improvement in steady states. Finally, we will apply the algorithms to the packet scheduling problem in wireless networks, and demonstrate that the distributed simplex scheduling can achieve dramatic steady state delay improvement as compared to distributed CSMA algorithms.

- Chapter 5 investigates suboptimal scheduling policies. We are particularly interested in the performance of maximal scheduling algorithms, which is easily amendable for distributed implementation due to its simplicity. We will formulate a lower bound on the throughput region with general PhyNets. We will also prove that maximal scheduling can achieve a certain fraction of the optimal throughput region. We then try to improve the performance of maximal scheduling for packet scheduling in wireless networks. In particular, we propose a static priority assisted maximal scheduling, and show that it can achieve significant improvement over maximal scheduling. We prove that the optimal static priority can be computed with low complexity.

1.4 RELATED WORK

The general scheduling framework proposed in this thesis is related to applications from a diverse range of research areas. In the literature, these problems have been analyzed assuming different models. In the sequel, we will provide a brief overview of the related work. Closely related results will be discussed in more detail in later chapters in the context of the each specific topic.

The physical factor graph modeling in Chapter 2 is closely related to the interference graph model in wireless networks. The interference graph model for packet scheduling in wireless network has been extensively investigated in the past [10, 14, 26, 46–50]. The construction of the inter-

ference graph depends heavily on the physical layer communication technology. For example, for spread spectrum communication systems such as Bluetooth and FH-CDMA networks, the interference graph is constructed based on the node exclusive interference model [10, 47], which specifies that any pair of transmitting links cannot share a common node. For the ubiquitous IEEE 802.11 networks, a two-hop interference model is commonly used [14, 48], which specifies that any pair of transmitting links must be separated by at least two hops, due to co-channel interference. A K-hop interference model was proposed in [49] to construct interference graph for general wireless networks, which generalizes the node exclusive model ($K = 1$) and 802.11 interference model ($K = 2$). Compared to these models in the literature, the contribution of this thesis is that we propose a hypergraph interference model [26], which not only preserves the graph structure, but also incorporates the cumulative effect of co-channel interference. The graph representation and interpretation of power flow coupling is well-known in power systems [51–53]. Recently, there have been growing research interests in investigating the design and performance analysis of optimal power flow algorithms that utilize the graph representation of the power system [54–56]. A physical graph-type representation of the thermal-aware work load scheduling in data centers was recently developed in [41]. We emphasize that the general physical factor graph model proposed in this thesis can include all such applications as special cases.

The augmented max-weight optimal scheduling schemes in Chapter 3 are motivated by the max-weight packet scheduling algorithm in wireless networks [11, 57, 58]. In the seminal work, [11] proposed a queue length weighted scheduling algorithm and proved its throughput optimality in multi-hop wireless networks. The max-weight algorithm was later generalized in [57, 58] to the scenario of cost-aware optimal scheduling. The 'pick-and-compare' algorithm was proposed in [12] to approximate the max-weight algorithm over multiple time slots, in order to reduce the computation overhead per time slot. Recently, there have been growing research interests in achieving distributed implementation of the max-weight algorithm using CSMA mechanisms [16, 59], which can be interpreted as applications of the Markov Chain Monte-Carlo (MCMC) methods with the interference

graph model [60]. The max-weight algorithm has also been investigated in the context of EV charging in power systems, mostly in a heuristic manner. In particular, [61] proposed a max-weight type EV charging algorithm and solved it using evolutionary algorithms without considering AC power flow constraints. In [62], a heuristic max-weight EV charging algorithm was implemented in a low voltage distribution system subject to voltage and congestion constraints. The max-weight algorithm has also been recently investigated in for the workload scheduling applications in data centers. [43] proposed a Lyapunov optimization based max-weight algorithm for the optimal admission control, routing and resource allocation in virtualized data centers. In [44], a two time scale max-weight algorithm was proposed for distributed routing and service management among geographically separated data centers. Compared to these algorithms, the augmented max-weight scheduling in this thesis not only generalizes the design to the scheduling in PhyNets, but also provides rigorous optimality guarantees with very mild assumptions on the stochastic dynamics.

The simplex scheduling algorithm in Chapter 4 is related to the centralized packet scheduling formulation in [63], which solves a static linear programming version of the packet scheduling in wireless networks. The distributed CSMA implementation in Chapter 4 in the context of wireless networks is closely related to the distributed CSMA algorithm design in [15, 16, 64, 65]. As will be shown later in this thesis, compared to such schemes, the simplex scheduling proposed in this thesis can achieve dramatic steady-state delay improvement in wireless networks.

The maximal scheduling algorithms investigated in Chapter 5 are motivated by the maximal packet scheduling algorithms in wireless networks. Maximal packet scheduling in wireless networks has been extensively investigated in the literature [14, 22, 47, 48]. [47] discussed the performance of maximal scheduling under the node exclusive interference model and demonstrated that it can achieve at least half of the optimal throughput region. [14, 48] investigated the throughput performance of maximal scheduling under a general interference graph model. [66, 67] discussed distributed implementations of the maximal scheduling algorithm. The maximal scheduling scheme in Chapter 5 is a generalization of such schemes from wireless networks to the generalized CPS,

with rigorous performance guarantees. The static priority based maximal scheduling in Chapter 5 is related to the longest queue first (LQF) scheduling in wireless networks, which dynamically assigns priority based on queue lengths. [68] considered the throughput performance of the LQF scheduling, and proposed a sufficient condition, which is called the 'local pooling' condition, for throughput optimality. The local pooling condition was later generalized to the 'local pooling factor' in [69], which corresponds to be the scheduling efficiency of LQF scheduling. There has been extensive research results [69–72] on estimating the local pooling factor with different interference models. Finally, distributed implementation of LQF scheduling are discussed in [71] and [73]. Compared to LQF scheduling, the static priority assisted maximal scheduling in this thesis achieves essentially the same bound on the 'local pooling factor' [69], while reduces the scheduling overhead associated with priority updates.

CHAPTER 2

THE SCHEDULING PROBLEM IN CYBER-PHYSICAL SYSTEMS

In this chapter, we propose a very general formulation of the scheduling problem in PhyNets. As described in Chapter 1, there are many scheduling-type problems in the literature, which have been modeled and analyzed independently in the context of different applications, such as packet scheduling in wireless networks, EV charging in smart grids, and workload scheduling in data centers. One contribution of this thesis is to show that these applications from diverse research domains can all be modeled and analyzed similarly, within one unified framework. We propose algorithms to solve the scheduling problem in Chapters 3-5.

The organization of this chapter is as follows. In Section 2.1 we propose the abstract physical factor graph model for scheduling application. Section 2.2 describes the queueing system model, and Section 2.3 proposes a mathematical formulation of the scheduling problem. Section 2.4 discusses how the formulation can be applied to many different CPS applications.

2.1 PHYSICAL FACTOR GRAPH

In an abstract manner, we assume that the physical plant of the CPS consists of a set $\mathcal{V}$ of *user nodes*, which have concrete physical meanings for the scheduling application. For example, for

packet scheduling in wireless networks, a user may correspond to a link, whereas for EV charging in power systems, a user refers to a bus in the power grid. The behavior of each user node $i \in \mathcal{V}$ is described by a set of variables $\{\alpha_i, \chi_i, s_i\}$, whose definitions are as follows. The first variable α_i is the *action variable*, which represents the operations that user i can perform. It will be shown later that the control action variable $\alpha_i(n)$ specifies the job departure of user i in each time slot n. For example, in wireless networks, $\alpha_i(n) \in \{0, 1\}$ can represent the transmission status of a link in time slot n, such that '1' represents transmitting, and '0' otherwise. We assume that each action variable α_i is *nonnegative* and lives in a *finite discrete* set, which we denote as $\mathcal{A}_i$. The main task of the scheduling problem is to optimally choose a sequence of actions of all user nodes $\{\alpha_i(n)\}$, subject to the physical and queue stability constraints, which we will describe very shortly. The physical constraint is represented by the *physical variable* χ_i, which lies in a feasible region $\mathcal{O}_i$. Each χ_i represents concrete physical quantities of interest for the scheduling problem. For example, for EV charging problems, χ_i may represent the voltage of each bus, whose magnitude has to lie in a bounded region $\mathcal{O}_i = [V_i^{\min}, V_i^{\max}]$. Finally, for general time-varying systems, we associate with each user i a *mode variable* s_i, which represents its 'local mode' and can change over time slots. For example, for the EV charging problem, the local mode may correspond to the non-EV household load. In wireless networks, the local mode of each link may correspond to the channel status, such as 'ON' and 'OFF'. We assume that each s_i takes value from a discrete set $\mathcal{S}_i$, and that both the set of control actions $\mathcal{A}_i$ and the set of feasible physical variables $\mathcal{O}_i$ are functions of s_i, as the available control actions and physical limits may change with the user's local mode.

The physical interactions of user nodes are modeled by a *physical factor graph* $\mathcal{G}(\mathcal{U}, \mathcal{F}, \mathcal{E})$. $\mathcal{U}$ is the set of variable nodes, which can be further partitioned according to different users as follows:

$$\mathcal{U} = \cup_{i \in \mathcal{V}} \{\alpha_i, \chi_i, s_i\}. \tag{2.1}$$

$\mathcal{F}$ is the set of factor nodes, each of which represents network coupling among the variable nodes,

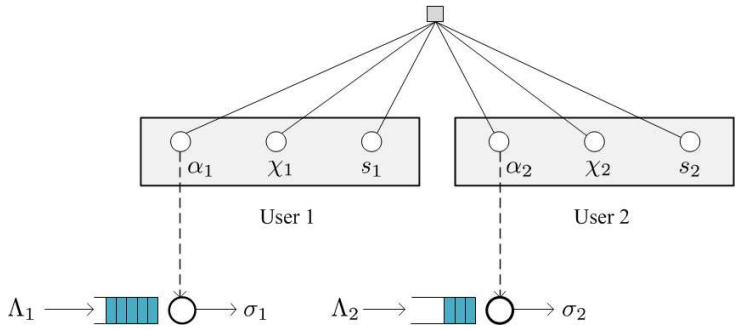


Figure 2.1: An example physical factor graph with its underlying queueing system. The white nodes represent the variable nodes, and the grey node represents the factor node.

as follows:

$$h_k(\alpha_{\mathcal{N}_k}, \chi_{\mathcal{N}_k}; s_{\mathcal{N}_k}) = 0, \forall k, \tag{2.2}$$

where h_k is an abstract function, and $\mathcal{N}_k$ is the set of users connected to the factor node $k \in \mathcal{F}$. Thus, (2.2) describes the network coupling of the control actions $\{\alpha_i\}$ and the physical variables $\{\chi_i\}$, due to certain physical laws. One example physical factor graph is shown in Fig. 2.1.

2.2 QUEUEING SYSTEM

We continue to formulate the queueing system model. As described in Chapter 1, 'scheduling' in this thesis refers to the coordinated actions of all users in the system, such that certain resources can be efficiently allocated among them to satisfy long term and uncertain demands. As the demand can be highly intermittent and uncertain, queueing systems are often used in the modeling, analysis and design for such problems. Here, the queue lengths represent the amount of 'backlogged demand', so that the desired throughput performance can be achieved in the presence of stochastic demand by stabilizing all queues. For the abstract scheduling problem formulation considered in this thesis, we refer to the demand as 'jobs'. We assume a time-slotted system, and associate each user node

$i \in \mathcal{V}$ with a queue. The queueing dynamics can be described as follows:

$$U_i(n) = U_i(n-1) - \sigma_i(n) + \Lambda_i(n). \tag{2.3}$$

In the above, $U_i(n)$ is the queue length of user i at the end of time slot n, $\sigma_i(n)$ is the number of job departures during time slot n, which is specified by a certain scheduling algorithm. $\Lambda_i(n)$ is the number of external stochastic job arrivals during time slot n. In this thesis, we impose very mild assumptions on the arrival processes:

Assumption 2.2.1. *$\Lambda_i(n)$ is uniformly bounded by a constant with probability 1 (w.p.1):*

$$\Lambda_i(n) \leq \Lambda_i^{\max}, \forall i, n, \tag{2.4}$$

where $\Lambda_i^{\max}$ is a positive constant. Further, $\Lambda_i(n)$ is subject to the Strong Law of Large Numbers (SLLN), as follows:

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \Lambda_i(n) = \lambda_i, \ \textit{w.p.1.}, \forall i, \tag{2.5}$$

where λ_i is the average job arrival rate for user i.

Notice that the above assumptions are very mild, as the arrival processes are allowed to be arbitrarily correlated across different time slots as well as different users. Thus, the model is very general, and can be used for many real world CPS applications.

The scheduling algorithm has to specify the time series of control actions $\{\alpha_i(n)\}$ to stabilize all queues. We assume that the job departure $\sigma_i(n)$ is related to the control action variable $\alpha_i(n)$ as follows

$$\sigma_i(n) = \alpha_i(n) \wedge U_i(n-1), \forall i, n. \tag{2.6}$$

In the above, $x \wedge y = \min(x, y)$, so that the queue lengths cannot become negative. We focus on the throughput performance of the scheduling schemes. Thus, 'stability' in this thesis refers to the *rate stability* [74]:

$$\lim_{n\to\infty} \frac{U_i(n)}{n} \to 0, \ \text{w.p.1}, \forall i. \tag{2.7}$$

Note that it is possible to obtain stronger stability results, such as positive recurrence of Markov chains [11, 75, 76], by placing more restricted assumptions on the stochastic arrival processes. This

will be addressed in future work. Section 3.2 provides an asymptotic result on time-average queue lengths under an augmented max-weight scheduling algorithm. Finally, notice that the action variables $\{\alpha_i(n)\}$ are always subject to the physical factor graph constraints as specified in the last section, which can be highly random, and vary across time slots, due to the randomness from the stochastic local modes $\{s_i(n)\}$. Similar to the arrival processes, we assume the following very mild assumptions on the statistics with the local mode variables:

Assumption 2.2.2. *The local mode processes $\{s_i(n)\}$ satisfy the following:*

$$\lim_{N\to\infty} \frac{\sum_{n=1}^{N} \mathbf{1}_{\{s(n)=s\}}}{N} = \pi_s, \textit{ w.p.1.} \tag{2.8}$$

where $\mathbf{1}_{\{\cdot\}}$ is the indicator function, i.e., $\mathbf{1}_{\{true\}} = 1$ and $\mathbf{1}_{\{false\}} = 0$, and π_s is the average time fraction that the system mode takes a particular value s.

In the next section we continue with the discussion by formulating the scheduling cost, and then, propose the general scheduling problem.

2.3 FORMULATION OF THE SCHEDULING PROBLEM

We assume the following general scheduling cost function at each time slot:

$$f(\alpha(n); s(n)) = \sum_{j\in\mathcal{J}} f_j(\alpha_{\mathcal{N}_j}(n); s_{\mathcal{N}_j}(n)), \tag{2.9}$$

where $\mathcal{J}$ can be interpreted as a set of *cost factor nodes*, and $\mathcal{N}_j$ is the set of user nodes associated with each cost factor node j. Thus, similar to the scheduling constraints, the cost function can also be decomposed in a graph manner. Note that this assumption can be made without loss of generality, since even a global cost function can be modeled as a factor node connected to all user nodes. We

are now ready to formulate the general cost-optimal scheduling problem:

$$\begin{aligned}
\textsf{SCH-C:} \min_{\{\alpha_i(n),\chi_i(n)\}} \ & \limsup_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N}\sum_{j\in\mathcal{J}} f_j(\alpha_{\mathcal{N}_j}(n); s_{\mathcal{N}_j}(n)) \\
\text{subject to} \ & U_i(n) = U_i(n-1) - \alpha_i(n) \wedge U_i(n-1) + \Lambda_i(n), \forall i \in \mathcal{V}, n \geq 1 \\
& h_k(\alpha_{\mathcal{N}_k}(n), \chi_{\mathcal{N}_k}(n); s_{\mathcal{N}_k}(n)) = 0, \forall k \in \mathcal{F}, n \geq 1 \\
& \chi_i(n) \in \mathcal{O}(s_i(n)), \forall i \in \mathcal{V}, n \geq 1 \\
& \alpha_i(n) \in \mathcal{A}_i(s_i(n)) \ \forall i \in \mathcal{V}, n \geq 1 \\
& \text{Stability of all queues} \qquad (2.10)
\end{aligned}$$

In words, we are interested in minimizing a long-term average scheduling cost, subject to the physical graph constraints in each time slot and the asymptotic queue stability constraints. The above scheduling formulation is very general, which includes many well-known applications as special cases. Further, it is very promising to develop distributed algorithms for such scheduling problems, due to the local physical graph specification of the constraints, which are represented by factor nodes.

We next formulate another version of the general scheduling problem. This corresponds to the case where certain knowledge about the optimal cost or budget information is available. For example, in power systems, it is typically assumed that the electricity cost can be estimated or predicted with good accuracy. In such cases, we can formulate the scheduling problem as the

following feasibility problem:

$$\begin{aligned}
\textsf{SCH-F}: \min_{\{\alpha_i(n),\chi_i(n)\}} \quad & 0 \\
\text{subject to} \quad & \limsup_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N} f_j(\alpha_{\mathcal{N}_j}(n); s_{\mathcal{N}_j}(n)) \le \hat{f}_j, \forall j \in \mathcal{J} \\
& U_i(n) = U_i(n-1) - \alpha_i(n) \wedge U_i(n-1) + \Lambda_i(n), \forall i \in \mathcal{V}, n \ge 1 \\
& h_k(\alpha_{\mathcal{N}_k}(n), \chi_{\mathcal{N}_k}(n); s_{\mathcal{N}_k}(n)) = 0, \forall k \in \mathcal{F}, n \ge 1 \\
& \chi_i(n) \in \mathcal{O}(s_i(n)), \forall i \in \mathcal{V}, n \ge 1 \\
& \alpha_i(n) \in \mathcal{A}_i(s_i(n)), \forall i \in \mathcal{V}, n \ge 1 \\
& \text{Stability of all queues}
\end{aligned} \tag{2.11}$$

where $\hat{f}_j$ is a budget or estimation of the optimal scheduling cost associated with cost factor node j. We will discuss solutions to both of the above scheduling problems SCH-C and SCH-F in later chapters. Before that, we first identify some important applications and show how they can be addressed by the general scheduling framework.

2.4 APPLICATIONS

In this section, we show that the general scheduling framework includes many CPS applications as special cases, such as packet scheduling in wireless networks, EV charging in smart grids, and workload scheduling in data centers. We start with the example of packet scheduling in wireless networks.

2.4.1 Packet Scheduling in Wireless Networks

We first introduce the packet scheduling problem in wireless networks, and then show that it is a special case of the general scheduling problem.

2.4.1.1 Introduction to Packet Scheduling in Wireless Networks

For packet scheduling in wireless networks, each user represents a link in the network. The queue at each link represents the currently back-logged packets waiting for transmission. The queueing dynamics can be described as follows:

$$U_i(n) = U_i(n-1) - \alpha_i(n) \wedge U_i(n-1) + \Lambda_i(n), \tag{2.12}$$

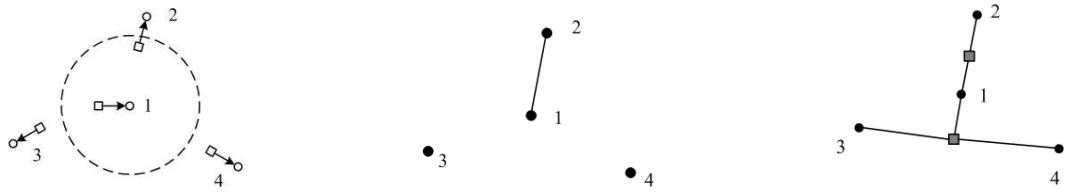
where $\Lambda_i(n)$ represents the number of arrived packets during time slot n, and the control action $\alpha_i(n) \in \{0, 1\}$ represents the transmission status of link i, such that $\alpha_i(n) = 1$ means that link i is transmitting, and that $\alpha_i(n) = 0$ means that link i remains silent.

We now describe the interference model. For simplicity of discussion, we assume that the wireless network is quasi-static, where the network topology remains constant for packet scheduling. For typical wireless networks, it is often assumed that a packet transmission for a link i is successful if its signal-to-interference-plus-noise ratio (SINR) is above a certain threshold:

$$\frac{P_i}{N_i + \sum_{j \in \sigma} I_{ji}} \geq \theta_i, \tag{2.13}$$

where P_i is the received signal power at link i, N_i is the noise power, I_{ji} is the received interference at link i from transmitting link j, and θ_i is the SINR threshold for successful packet reception for link i, which is determined by the physical layer modulation, detection and coding specifications. Note that for the simplicity of notation, we denote σ as the set of transmitting links, so that $j \in \sigma$ implies that link j is a transmitting link.

The SINR model can accurately describe the interference constraints in wireless networks. However, it is very difficult for the design of distributed scheduling algorithms, due to its global nature. For typical wireless networks, in particular wireless ad hoc networks, where a central scheduling entity often does not exist, it is crucial to develop an interference model that allows design and analysis of distributed scheduling algorithms. In the literature, this is achieved by the interference graph model [10, 27, 58], which models the interference as binary. The interference



(a) (b) (c)

Figure 2.2: (a) A sample wireless network with 4 links, where square nodes are the transmitters, and round nodes are the receivers. (b) Its graph interference model. (c) Its hypergraph interference model.

graph specifies that the transmission of a particular link fails if and only if there is a concurrent transmission of any neighboring link. For example, consider the 4-link wireless network in Fig. 2.2 (a), where an interference graph can be constructed by, for example, placing a guard zone [50] with certain radius around the receiver of each link. Two links form an edge in the interference graph if one's transmitter is in the guard zone associated with the other. In such a case, the interference graph for Fig. 2.2 (a) has only one edge $e = \{1, 2\}$, as shown in Fig. 2.2 (b). Therefore, a transmission schedule is valid as long as links 1 and 2 are not transmitting simultaneously.

We next introduce a hypergraph interference model described by our recent work [26]. The motivation is that the interference graph is a rigid model, which over-simplifies the physical interference in typical wireless networks [77, 78], since it does not take into account the *cumulative* effect of interference. That is, the transmission failure at a link may occur due to the sum interference from concurrent transmitting links, even though the contribution from each link is small. For example, in Fig. 2.2 (a), it is possible that link 1 fails when links $\{1, 3, 4\}$ are scheduled, due to the sum interference from both link 3 and link 4. In such a case, the interference graph can only guarantee that the transmission at link 1 is successful when only one of the other two links is transmitting, due to its binary nature. On the other hand, if one builds the graph conservatively by increasing the size of the guard zone, such that two additional edges $\{1, 3\}$ and $\{1, 4\}$ are included (note that both link 3 and link 4 have the same distance to link 1 in this example), the network capacity is reduced,

because when link 1 transmits, neither link 2 nor link 3 is allowed to transmit, even though there is no collision if only one of them transmits.

Realizing the inaccuracy of the graph model, we proposed a *hypergraph* interference model, which not only considers the cumulative nature of co-channel interference in wireless networks, but also is easy for distributed implementation, due to the local construction. The detailed construction procedure is as follows. The key observation is that, for typical wireless networks, a major portion of the total interference is contributed by only a few nearby transmitting links. Thus, we can approximate the SINR locally with very good accuracy as

$$\frac{P_i}{N_i + \sum_{j\in\sigma} I_{ji}} \approx \frac{P_i}{N_i + \sum_{j\in\sigma}(I_{ji}\cdot \mathbf{1}_{\{j\in\mathcal{L}_i\}})}, \tag{2.14}$$

where $\mathcal{L}_i$ is the set of 'local' links around link i:

$$j \in \mathcal{L}_i \text{ if } \frac{S_i}{N_i + I_{ji}} < \beta_i, \tag{2.15}$$

where β_i is a properly chosen threshold. Based on the above SINR approximation, we can construct a hyperedge $e = \{i, i_1, i_2, \ldots, i_{k-1}\}$ if

$$\frac{P_i}{N_i + \sum_{s=1}^{k-1} I_{i_s i}} < \theta_i. \tag{2.16}$$

where the links $\{i_1, i_2, \ldots, i_{k-1}\}$ are selected only if they are in $\mathcal{L}_i$, so that the MAC coordination can be restricted to only local links. This simply implies that the links $\{i, i_1, i_2, \ldots, i_{k-1}\}$ are not allowed to transmit simultaneously, since link i will fail due to (2.16). Fig. 2.2 (c) shows the hypergraph interference model corresponding to the 4-link wireless network. Notice the new hyperedge $e = \{1, 3, 4\}$, due to the fact that the cumulative interference from links 3 and 4 can cause link 1 to fail. Finally, note that by adjusting $\{\beta_i\}$ and the maximum allowed cardinality of all hyperedges, the interference accuracy can be gradually improved from the binary interference graph model (small β_i and $\max|e| = 2$, where $|\cdot|$ denotes the cardinality) to the accurate SINR model (large β_i and $\max|e| = |\mathcal{V}|$). We provide quantitative analysis and simulation results on the

accuracy of the hypergraph interference model in Appendix A.

2.4.1.2 Scheduling Formulation for Packet Scheduling in Wireless Networks

We first demonstrate that the hypergraph interference model can be converted to the physical graph model for the general scheduling problem, as follows. For each link-hyperedge pair (i, e) such that link $i \in e$, we add χ_i^e to the set of physical variables of link i, and a factor node for the following network coupling:

$$\chi_i^e = \sum_{l \in e} \alpha_l. \tag{2.17}$$

Thus, χ_i^e represents the link i's local copy about the total number of transmitting links in the set e, which can be interpreted as an estimate of the interference level for the set of links in e. The feasible region $\mathcal{O}_i^e$ is as follows:

$$\mathcal{O}_i^e = \{\chi_i^e : 0 \leq \chi_i^e \leq |e| - 1\}. \tag{2.18}$$

Thus, one can easily observe that the above factor graph model is equivalent to the hypergraph interference model. Further, the general queueing model in (2.3) can be naturally applied to the packet queues in (2.12) for wireless networks, and the general scheduling cost function in (2.9) can also be well adapted to model typical scheduling costs in wireless networks, such as average transmission power. Thus, we conclude that the general scheduling problem formulation includes the packet scheduling in wireless networks as a special case. We write the packet scheduling in wireless networks as below, for completeness:

$$\begin{aligned}
\min_{\{\alpha_i(n), \chi_i(n)\}} \ & \limsup_{N \to \infty} \frac{1}{N} \sum_{n=1}^{N} \sum_{j \in \mathcal{J}} f_j(\alpha_{\mathcal{N}_j}(n)) \\
\text{subject to } \ & U_i(n) = U_i(n-1) - \alpha_i(n) \wedge U_i(n-1) + \Lambda_i(n), \forall i \in \mathcal{V}, n \geq 1 \\
& \chi_i^e(n) = \sum_{l \in e} \alpha_l(n), \forall (i, e) \text{ with } i \in e, n \geq 1 \\
& 0 \leq \chi_i^e(n) \leq |e| - 1, \forall (i, e) \text{ with } i \in e, n \geq 1 \\
& \alpha_i(n) \in \{0, 1\}, \forall i \in \mathcal{V}, n \geq 1 \\
& \text{Stability of all queues}
\end{aligned} \tag{2.19}$$

2.4.2 EV Charging in Power Systems

As another important application, we show that the coordinated EV charging problem in power systems can be included as a special case of the general CPS scheduling problem.

2.4.2.1 Introduction to EV Charging in Power Systems

For the EV charging application, each user $i \in \mathcal{V}$ represents a bus in the power system. We assume that each bus is either associated with one EV, or is not associated with any EV at all. Such a model is used to represent the residential charging scenario, where the owner of a household either uses EV for daily commute or does not own any EV. An example of the system model is shown in Fig. 2.3. In this case, the 'jobs' correspond the amount of energy needed to fully refill the battery of each EV. For example, for an EV at bus i with a battery of 10kWh capacity and 60% state of charge (SoC), the corresponding energy queue length is 4kWh. The dynamics of the EV queue length is as follows:

$$U_i(n) = U_i(n-1) - \alpha_i(n) \wedge U_i(n-1) + \Lambda_i(n),\ \forall i, n. \tag{2.20}$$

In above, the control action $\alpha_i(n)$ can be further expressed as follows:

$$\alpha_i(n) = \eta_i P_i(n) \Delta t \tag{2.21}$$

where η_i is charging circuit efficiency of the EV at bus i, Δt is the length of a time slot, and $P_i(n)$ is the active charging power of EV i. It is assumed that $P_i(n)$ belongs to a finite set of charging rates, which we denote as $\mathcal{P}_i$. $\Lambda_i(n)$ is the amount of external 'energy job' arrivals during time slot n, due to the energy consumption from driving. Note that for any bus without EV, the corresponding energy queue length $U_i(n)$ is trivially zero all the time and the control action $\alpha_i(n)$ is also always zero.

Given the above queueing model, the goal of the EV charging scheduler is to specify the time series of charging power $\{P_i(n)\}$, so that a long-term average charging cost is minimized, while ensuring that the energy needs of all EV are successfully satisfied. Note that it is the general cost

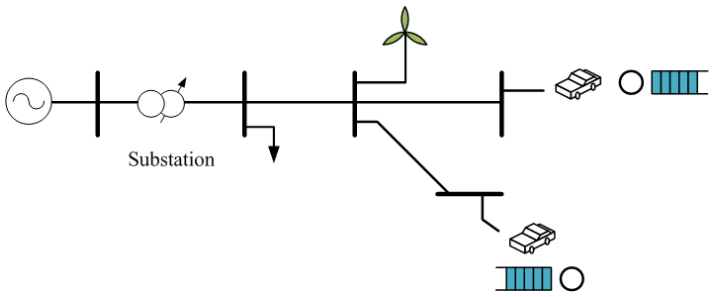


Figure 2.3: An example power system with EV charging application.

function in (2.9) can be well used to model typical average charging cost functions, such as the ones based on electricity prices. Thus, it is now sufficient to show that the physical charging constraints can be described by in a physical factor graph manner. The charging constraints are as follows. Firstly, the charging power $P_i(n)$ for each bus i is subject to the charging circuit rating constraint:

$$P_i^{\min} \leq P_i(n) \leq P_i^{\max}, \forall i. \tag{2.22}$$

Further, the charging process is also constrained by the EV availability, so that

$$P_i(n) = 0 \text{ if } a_i(n) = 0, \forall i, \tag{2.23}$$

where $a_i(n)$ the indicator function that EV i is 'available' for charging during time slot n, i.e., $a_i(n) = 1$ if the EV is available for charging, and $a_i(n) = 0$ otherwise. Notice that $\{a_i(n)\}$ is an external random process, which depends on the stochastic EV driving patterns. Further, the impact of the EV charging power to the power system states can be modeled by the following AC power flow equations:

$$P_i^{\text{net}}(n) + P_i(n) = -V_i(n) \sum_{j \in \mathcal{N}_i} V_j(n)[G_{ij} \cos(\theta_{ij}(n)) + B_{ij} \sin(\theta_{ij}(n))] \tag{2.24}$$

$$Q_i^{\text{net}}(n) = -V_i(n) \sum_{j \in \mathcal{N}_i} V_j(n)[G_{ij} \sin(\theta_{ij}(n)) - B_{ij} \cos(\theta_{ij}(n))]. \tag{2.25}$$

In above, $V_i(n)$ is the voltage magnitude at bus i at time slot n, and

$$\theta_{ij}(n) = \theta_i(n) - \theta_j(n) \tag{2.26}$$

is the voltage phase angle difference between bus i and j during time slot n. G_{ij} and B_{ij} are the conductance and susceptance of the transmission line between bus i and its neighboring bus j, respectively. $P_i^{\text{net}}(n)$ and $Q_i^{\text{net}}(n)$ are the net active and reactive power consumption for the non-EV load, as follows:

$$P_i^{\text{net}}(n) = P_i^{\text{base}}(n) - P_i^{\text{renew}}(n) \tag{2.27}$$

$$Q_i^{\text{net}}(n) = Q_i^{\text{base}}(n) - Q_i^{\text{renew}}(n), \tag{2.28}$$

where $P_i^{\text{renew}}(n)$ and $Q_i^{\text{renew}}(n)$ correspond to the active and reactive distributed generation with renewable energy sources at bus i, respectively. One example is the wind generator in Fig. 2.3. Notice that both $P_i^{\text{renew}}(n)$ and $Q_i^{\text{renew}}(n)$ are trivially zero if bus i has no renewable generation. Finally, the voltage of each bus in the system has the following voltage limits:

$$V_i^{\min} \leq V_i(n) \leq V_i^{\max}, \forall i. \tag{2.29}$$

Thus, if the charging processes of EVs are uncoordinated, it is well possible that the EV charging at one bus can make the voltage constraint at a remote bus become violated. On the other hand, if the charging processes of all EVs are coordinated carefully, it is very promising that not only the power system can operate reliably, but also the highly intermittent renewable energy sources can be successfully 'absorbed' to refill the EV batteries.

2.4.2.2 Scheduling Formulation for EV Charging

We now show that the above EV charging problem can be included in the general scheduling formulation. It is easy to verify that the EV battery dynamics in (2.20) is a special case of the general queueing model. We only need to show that the physical constraints can be modeled by a

factor graph. For each bus i, define the local mode variable as

$$s_i = (a_i, P_i^{\text{net}}, Q_i^{\text{net}}), \tag{2.30}$$

and physical variable as

$$\chi_i = (V_i, \theta_i). \tag{2.31}$$

Thus, it is easy to verify that the constraints in (2.22) and (2.23) can be easily modeled by the feasible region $\alpha_i \in \mathcal{A}_i(s_i)$. Further, the voltage limit in (2.29) can be modeled by the region $\chi_i \in \mathcal{O}_i$. Notice that in this case, the region $\mathcal{O}_i$ does not depend on the mode variable s_i. Finally, we can associate a physical factor node with each of the AC power flow equation in (2.25). Therefore, we conclude that the EV charging problem can be modeled as a special case of the general CPS scheduling problem. For completeness, we write the EV charging problem below:

$$\begin{aligned}
\min_{\{P_i(n),V_i(n),\theta_i(n)\}} \quad & \limsup_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N}\sum_{j\in\mathcal{J}} f_j(P_{\mathcal{N}_j}(n); a_{\mathcal{N}_j}(n), P^{\text{net}}_{\mathcal{N}_j}(n), Q^{\text{net}}_{\mathcal{N}_j}(n)) \\
\text{subject to} \quad & U_i(n) = U_i(n-1) - (P_i(n)\eta_i\Delta t) \wedge U_i(n-1) + \Lambda_i(n), \forall i \in \mathcal{V}, n \\
& P_i^{\text{net}}(n) + P_i(n) = -V_i(n)\sum_{j\in\mathcal{N}_i} V_j(n)[G_{ij}\cos(\theta_{ij}(n)) + B_{ij}\sin(\theta_{ij}(n))], \forall i, n \\
& Q_i^{\text{net}}(n) = -V_i(n)\sum_{j\in\mathcal{N}_i} V_j(n)[G_{ij}\sin(\theta_{ij}(n)) - B_{ij}\cos(\theta_{ij}(n))], \forall i, n \\
& V_i^{\min} \le V_i(n) \le V_i^{\max}, \forall i, n \\
& P_i(n) \in \mathcal{P}_i, \forall i, n \\
& P_i(n) = 0 \text{ if } a_i(n) = 0, \forall i, n \\
& \text{Stability of all queues}
\end{aligned} \tag{2.32}$$

where we have written $\theta_{ij} = \theta_i - \theta_j$ as an abbreviation for notation simplicity.

2.4.3 Workload Scheduling in Data Centers

Finally, we will show that the general scheduling formulation can include the workload scheduling in data centers as a special case.

2.4.3.1 Introduction to Workload Scheduling in Data Centers

We focus on the thermal-aware computing resource allocation problem within one data center [40]. In this case, each user corresponds to a server in a data center. A user i is associated with a queue of computing tasks, where the queue length represents the amount of computing tasks to be processed. The queueing dynamics is as follows:

$$U_i(n) = U_i(n-1) - g_i(v_i(n)) \wedge U_i(n-1) + \Lambda_i(n),\ \forall i, n, \tag{2.33}$$

where $v_i(n)$ is the processor speed, and $g_i(\cdot)$ is a mapping between the processor speed to the computing task processing rate. $\Lambda_i(n)$ is the computing task arrival process, which is external and random. Thus, each user i has to dynamically adjust the speed of the processor v_i to ensure that the computing tasks can be successfully finished. We further assume that v_i belongs to a finite set of feasible speeds, which we denote as $\mathcal{A}_i$. An example workload scheduling is shown in Fig. 2.4.

A naive solution would be to set $v_i(n) = v_i^{\max}$ for each server $i \in \mathcal{V}$ to maximize the processing speed. However, such control actions is in general infeasible, due to the thermal limit constraints, as follows. Firstly, the temperature of each processor i is subject to the following limit:

$$T_i(n) \leq T_i^{\max}, \forall i, n, \tag{2.34}$$

where $T_i^{\max}$ is the maximum allowed operational temperature specified by the device manufacturer. Thus, the speed of a processor i has to be judiciously adjusted to avoid hardware failures and reliability issues. Secondly, the temperatures of different processors are coupled. This is because the power dissipation of one processor will increase the local temperature, which will also affect the temperature at other processors. The relationship among the heat transfer between different locations can be derived following the law of energy conservation [40]. For example, a linearized thermal model for such coupling in [40] is as follows:

$$T_i(n) = T_i^{\text{amb}}(n) + \sum_{j \in \mathcal{N}_i} d_{ij}\phi_j(v_j(n)), \tag{2.35}$$

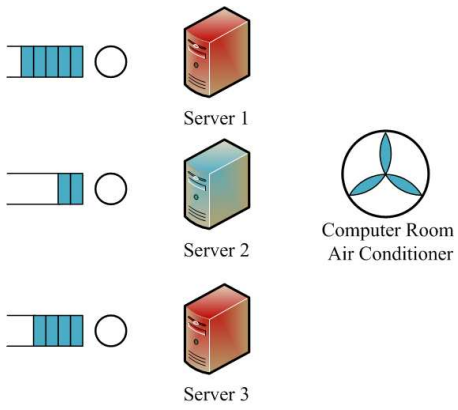


Figure 2.4: An example of work load scheduling in data centers, where the color of each server illustrates its temperature.

where $T_i^{\text{amb}}(n)$ is the ambient temperature, which is random, due to the stochastic dynamics of the cooling devices, such as the computer room air conditioner (CRAC) in Fig. 2.4. $\{d_{ij}\}$ are the heat distribution coefficients, and $\phi_j(\cdot)$ is the power dissipation function, which is a mapping from the processor speed v_i to its power dissipation. A commonly adopted power dissipation function is the 'cube' model, where $\phi_i(v_i) = c_i v_i^3$ is proportional to the cube of the processor speed [79].

2.4.3.2 Scheduling Formulation for Data Center Workload Scheduling

We now show that the above workload scheduling problem can be included as a special case of the general scheduling framework. It is easy to see that the abstract queueing model in Section 2.2 easily applies to the case of computing tasks. Thus, it is sufficient to show that the physical constraints can be modeled by a factor graph. Note that for each user i, we can define its control variable $\alpha_i = v_i$, local mode variable as $s_i = T_i^{\text{amb}}$, and the physical variable as $\chi_i = T_i$. Thus, it is easy to see that $\mathcal{A}_i$ and $\mathcal{O}_i$ correspond to the feasible set of processor speed and temperature limits, respectively. Further, we can associate a factor node k with each equality in (2.35), which represents the network coupling between the control actions and the physical variables. For completeness, we

write the data center workload scheduling problem below:

$$\begin{aligned}
\min_{\{v_i(n),T_i(n)\}} \ & \limsup_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N}\sum_{j\in\mathcal{J}} f_j(v_{\mathcal{N}_j}(n); T^{\text{amb}}_{\mathcal{N}_j}(n)) \\
\text{subject to } \ & U_i(n) = U_i(n-1) - g_i(v_i(n)) \wedge U_i(n-1) + \Lambda_i(n), \forall i, n \\
& T_i(n) = T_i^{\text{amb}}(n) + \sum_{j\in\mathcal{N}_i} d_{ij}\phi_j(v_j(n)), \forall i, n \\
& T_i(n) \le T_i^{\max}, \forall i, n \\
& v_i(n) \in \mathcal{A}_i, \forall i, n \\
& \text{Stability of all queues}
\end{aligned} \tag{2.36}$$

where the scheduling cost function may correspond to the power consumption.

CHAPTER **3**

OPTIMAL SCHEDULING IN THE DYNAMIC REGIME: AUGMENTED MAX-WEIGHT SCHEDULING

In Chapter 2, we proposed a general abstract scheduling problem for PhyNets and demonstrated that it includes CPS applications from diverse research areas. This chapter tries to solve the scheduling problem optimally using augmented max-weight algorithms, which generalize the max-weight scheduling algorithm in [11, 57] for wireless networks. This chapter focuses on the dynamic regime, where the local mode processes $\{s_i(n)\}$ are stochastic and vary over time slots. In Chapter 4, we will focus on the quasi-static regime, where the local modes remain constant for the scheduling application, and show that a simplex scheduling algorithm can be applied with improved performance.

We propose three augmented max-weight algorithms in this chapter. The first one is Algorithm 3.1.1, which computes a max-weight schedule in each time slot with virtual cost queues. The second one is Algorithm 3.1.2, which can be interpreted as a 'pick-and-compare' implementation of the first algorithm. The thrid one is Algorithm 3.2.1, which does not assume knowledge about the optimal scheduling cost by adopting a Lyapunov optimization approach [80] to compute a schedule in each time slot that maximizes a queue length weighted departure minus the instantaneous scheduling cost. All algorithms proposed in this chapter are amendable for distributed implementations, due to the physical factor graph representation of the scheduling constraints. However, the specific

implementation method will depend on the structure of each CPS application. For example, the max-weight algorithms in Algorithm 3.1.1 can be implemented in a distributed manner for packet scheduling in wireless networks using the distributed CSMA algorithms [15, 16], which can be interpreted as applications of Markov Chain Monte-Carlo methods. In power systems, the max-weight algorithms can be implemented by distributed optimal power flow algorithms [56], which can be interpreted as applications of the dual decomposition methods. Finally, Algorithm 3.1.2 allows much easier distributed implementation than the other two algorithms, as it only requires the random generation of a new schedule and comparison against an old schedule. Such a scheme can be easily implemented in a distributed manner using average consensus algorithms.

The organization of this chapter is as follows. In Section 3.1 we propose augmented max-weight algorithms for the feasibility problem SCH-F in Chapter 2, which assumes an estimate of the optimal cost or budget information, and proves stability results using fluid limits. In Section 3.2 we propose an augmented max-weight algorithm for the optimization problem SCH-C and proves its optimality using Lyapunov drift analysis. Section 3.3 demonstrates the performance of the augmented max-weight scheme for the important application of coordinated EV charging in power systems.

3.1 AUGMENTED MAX-WEIGHT SCHEDULING WITH COST KNOWLEDGE

In this section, we propose max-weight scheduling algorithms to solve the feasibility problem SCH-F. We remind the reader that SCH-F assumes estimation of the optimal scheduling cost or scheduling budget information, and requires the scheduler to satisfy the asymptotic scheduling cost bound. The algorithms proposed in this section can be regarded as augmentations of the conventional max-weight algorithm [11], in that a novel virtual queue mechanism is introduced to achieve cost-aware optimal scheduling. In particular, Algorithm 3.1.1 is a direct generalization of the max-weight algorithm in [11], whereas Algorithm 3.1.2 can be regarded as an amortized version of the max-weight algorithm, by randomly 'picking-and-comparing' schedules to approximate the max-

weight schedule over a long time interval, in order to reduce the computation in each time slot and achieve distributed implementation. We will prove the optimality results using the technique of fluid limits, to guarantee the sample path based cost optimality and stability with very mild assumptions on the stochastic dynamics and cost estimation processes. We start with the model of virtual cost queues.

3.1.1 Virtual Cost Queue

We associate with each component of the cost function $f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j})$ an estimation process $\{\hat{f}_j(n)\}$. The only assumption on $\{\hat{f}_j(n)\}$ is the following:

$$f_j^\star \leq \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \hat{f}_j(n) \leq f_j^\star + \epsilon_j, \text{ w.p.1}, \tag{3.1}$$

where ϵ_j is a positive constant, and

$$f_j^\star = \limsup_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} f_j(\alpha_{\mathcal{N}_j}^\star(n); s_{\mathcal{N}_j}(n)) \tag{3.2}$$

can be interpreted as the contribution of factor node $f_j(\cdot)$ to the optimal cost. $\{\alpha_{\mathcal{N}_j}^\star(n)\}$ is a solution of the cost-aware optimal scheduling problem SCH-C. For simplicity of notation, define

$$\hat{f}_j^\star = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \hat{f}_j(n) \tag{3.3}$$

as the estimated average optimal cost. Thus, we require that the scheduling algorithm cannot incur any asymptotic average cost larger than $\hat{f}_j^\star$ for the cost factor node $f_j(\cdot)$.

The key in achieving such guarantee is to introduce a virtual cost queue for each component of the cost function $f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j})$, which we denote as $\Phi_j(n)$. The queueing dynamics of $\Phi_j(n)$ is as follows:

$$\Phi_j(n) = \Phi_j(n-1) - \hat{f}_j(n) \wedge \Phi_j(n-1) + f_j(\alpha_{\mathcal{N}_j}(n); s_{\mathcal{N}_j}(n)). \tag{3.4}$$

Fig. 3.1. shows one example virtual queue. The instantaneous scheduling cost in each time slot $f_j(\alpha_{\mathcal{N}_j}(n); s_{\mathcal{N}_j}(n))$ can be interpreted as the arrival process to the virtual queue, whereas the esti-

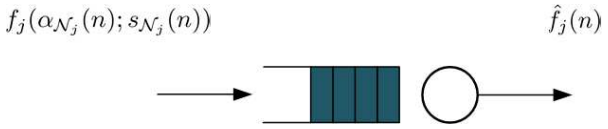


Figure 3.1: An example virtual cost queue.

mated scheduling cost $\hat{f}_j(n)$ corresponds to the instantaneous departure of the virtual queue. Thus, intuitively, if the virtual cost queue is rate stable, the average arrival rate has to be the same as the average departure rate, which is at most ϵ_j from the optimal cost, due to (3.1). This will be proved rigorously by fluid limits later. We next describe the augmented max-weight scheduling algorithm.

3.1.2 Augmented Max-Weight Scheduling Algorithms

We first propose a direct augmentation of the max-weight scheduling algorithm in Algorithm 3.1.1. One important feature of the scheduling algorithm is that it is *myopic*, which computes the schedules in each time slot only using the queue lengths in the current time slot. That is, according to (3.5), the scheduling algorithm always tries to stabilize the job and virtual queues by maximizing a queue length weighted job departures, penalized by the virtual queue lengths weighted arrivals. Compared to the conventional max-weight algorithm [11], the new component is the penalization term induced by the virtual queue lengths, due to the incorporation of scheduling cost. Thus, when the past scheduling decisions incur higher than expected cost, the virtual cost queues become large, which discourages the scheduling algorithm from choosing high cost schedules, and vice versa. Finally, the trade-off between minimizing queue lengths and scheduling cost can be adjusted by the constant β, which can be chosen by system specification and historical data.

We next present a 'pick-and-compare' version of the augmented max-weight scheduling algorithm in Algorithm 3.1.2, which can be regarded as a generalization of the algorithm in [11]. In the algorithm, the function $w(\cdot; n)$ corresponds to the queue lengths weighted departure, which is

Algorithm 3.1.1 Augmented Max-Weight Scheduling

1: For each time slot n, compute $\alpha(n)$ by solving the following:

$$\begin{aligned} \text{maximize}_{\{\alpha_i,\chi_i\}} \quad & \sum_{i\in\mathcal{V}} U_i(n-1)\alpha_i - \beta\sum_{j\in\mathcal{J}}\Phi_j(n-1)f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}(n)) \\ \text{subject to} \quad & h_k(\alpha_{\mathcal{N}_k}, \chi_{\mathcal{N}_k}; s_{\mathcal{N}_k}(n)) = 0, \forall k \in \mathcal{F} \\ & \chi_i \in \mathcal{O}(s_i(n)), \forall i \in \mathcal{V} \\ & \alpha_i \in \mathcal{A}_i(s_i(n)), \forall i \in \mathcal{V} \end{aligned} \tag{3.5}$$

2: Update queues $\{U_i(n)\}$ and virtual queues $\{\Phi_j(n)\}$ according to (2.3) and (3.4), respectively.

defined as follows:

$$w(\alpha; n) = \sum_{i\in\mathcal{V}} U_i(n-1)\alpha_i - \beta\sum_{j\in\mathcal{J}}\Phi_j(n-1)f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}(n)), \tag{3.6}$$

and the schedule $\alpha^{\text{old}}(s)$ is defined as the last chosen schedule when the system mode is at s. Thus, Algorithm 3.1.2 first randomly 'pick' a schedule α', and compare it with the $\alpha^{\text{old}}(s(n))$, which is the latest schedule under the system mode $s(n)$. The algorithm then chooses the one with the larger weight. Notice that such scheme needs to store the 'old' schedules $\alpha^{\text{old}}(s(n))$, which may require certain amount of memory. On the other hand, the algorithm can be well implemented in certain CPS applications where the total number of system modes is small, or the system modes remain constant for the scheduling application.

The above 'pick-and-compare' algorithm belongs to the category of the augmented max-weight scheduling in that it can be regarded as computing the max-weight schedule in an approximately 'simulated annealing' fashion [12], so that the schedules are gradually improved towards the max-weight solution. Thus, compared to the direct augmented max-weight approach in Algorithm 3.1.1, the 'pick-and-compare' Algorithm 3.1.2 can substantially reduce the computation per time slot. For example, for packet scheduling in wireless networks, Algorithm 3.1.1 corresponds to the max-weight independent set (MWIS) problem, which is well-known to be NP-hard. On the other hand, Algorithm 3.1.2 has low complexity, since it only requires a random independent set generation and comparison. Further, Algorithm 3.1.2 is easily amendable for distributed implementation, such as

Algorithm 3.1.2 Augmented Max-Weight Scheduling: Pick-and-Compare

For each time slot n, randomly generate α', such that

$$\mathbb{P}(\alpha' = \alpha) \geq \epsilon_0 \tag{3.7}$$

for any $\alpha \in \mathcal{C}(s)$.
if $w(\alpha'; n) > w(\alpha^{\text{old}}(s(n)); n)$ **then**
  $\alpha(n) = \alpha'$;
  $\alpha^{\text{old}}(s(n)) = \alpha'$;
else
  $\alpha(n) = \alpha^{\text{old}}(s(n))$;
end if
Update queues $\{U_i(n)\}$ and virtual queues $\{\Phi_j(n)\}$ according to (2.3) and (3.4), respectively.

using average consensus for the 'compare' phase. This can dramatically reduce the coordination overhead and simplify system design for CPS applications.

3.1.3 Optimality Proof

We next prove the optimality of the above scheduling algorithms, which is stated in the following theorem:

Theorem 3.1.1. *Assume that the problem SCH-F is feasible with $\{\hat{f}_j^\star\}$, and that $\{\hat{f}_j^\star\}$ satisfies (3.1). The following holds for the augmented max-weight scheduling schemes in Algorithm 3.1.1 and Algorithm 3.1.2:*

$$\limsup_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \sum_{j\in\mathcal{J}} f_j(\alpha_{\mathcal{N}_j}(n); s_{\mathcal{N}_j}(n)) \leq f^\star + \epsilon, \textit{w.p.1}, \tag{3.8}$$

where $f^\star$ is the optimal scheduling cost for SCH-C, and

$$\epsilon = \sum_{j\in\mathcal{J}} \epsilon_j. \tag{3.9}$$

Further, all job queues are rate stable.

The above theorem guarantees the asymptotic optimality of the augmented max-weight scheduling algorithm, in the sense that (3.8) holds for any gap $\epsilon > 0$ on the scheduling cost. Notice that we assume that the scheduling cost is estimated in an entirely *online* manner, by adopting the novel virtual queue technique. Thus, we only require that the ϵ-gap hold asymptotically. Such mild as-

sumption can substantially simplify the design and analysis in certain CPS applications, where the optimal scheduling cost is hard to obtain initially, and thereby can only be obtained in an online manner.

We next prove the theorem. For the ease of demonstration, we need to first simplify some notations and present a compact formulation of the queueing system.

3.1.3.1 A Reformulation of the Queueing System

For a fixed system mode $s \in \mathcal{S}$, we denote the set of feasible control actions as $\mathcal{C}(s)$. This is a compact representation of the set of feasible control actions $\{\alpha_i\}$ which satisfy the physical factor graph constraints in (3.5). Denote $T_s^\alpha(n)$ as a counting process which represents the total number of time slots that a control action α is chosen when the system mode is s during the first n time slots. We can rewrite the queueing dynamics in a very compact form as follows:

$$U_i(n) = U_i(0) - \sum_{s\in\mathcal{S}} \sum_{\alpha\in\mathcal{C}(s)} \alpha_i T_s^\alpha(n) + \Lambda_i(n) + Y_i(n), \forall i \in \mathcal{V} \tag{3.10}$$

$$\Phi_j(n) = \Phi_j(0) + \sum_{s\in\mathcal{S}} \sum_{\alpha\in\mathcal{C}(s)} f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}) T_s^\alpha(n) - \hat{F}_j(n) + Z_j(n), \forall j \in \mathcal{J} \tag{3.11}$$

$$\sum_{\alpha\in\mathcal{C}(s)} T_s^\alpha(n) = T_s(n), \forall s \in \mathcal{S} \tag{3.12}$$

$$\sum_{s\in\mathcal{S}} T_s(n) = n, \tag{3.13}$$

$$T_s^\alpha(n) \text{ is non-decreasing, } \forall s \in \mathcal{S}, \alpha \in \mathcal{C}(s), \tag{3.14}$$

where $\hat{F}_j(n)$ can be written as follows:

$$\hat{F}_j(n) = \sum_{\tau=1}^{n} \hat{f}_j(\tau). \tag{3.15}$$

$Y_i(n)$ and $Z_j(n)$ are system 'idling processes' that prevent the queues from becoming negative. $T_s(n)$ is the total number of time slots that the system is in mode s, according to the definition in (3.12). Thus, (3.13) follows naturally, since the system has to be in one mode during each time slot.

3.1.3.2 Fluid Limits

The proof is done by the technique of fluid limits, which is a general framework in analyzing stochastic systems. A brief introduction of fluid limits is in Appendix B.1. The queueing system in the fluid limit is as follows:

$$\bar{U}_i(t) = \bar{U}_i(0) - \sum_{s\in\mathcal{S}}\sum_{\alpha\in\mathcal{C}(s)} \alpha_i \bar{T}_s^{\alpha}(t) + \bar{\Lambda}_i(t) + \bar{Y}_i(t), \forall i \in \mathcal{V} \tag{3.16}$$

$$\bar{\Phi}_j(t) = \bar{\Phi}_j(0) + \sum_{s\in\mathcal{S}}\sum_{\alpha\in\mathcal{C}(s)} f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j})\bar{T}_s^{\alpha}(t) - \bar{F}_j(t) + \bar{Z}_j(t), \forall j \in \mathcal{J} \tag{3.17}$$

$$\sum_{\alpha\in\mathcal{C}(s)} \bar{T}_s^{\alpha}(t) = \bar{T}_s(t), \tag{3.18}$$

$$\sum_{s\in\mathcal{S}} \bar{T}_s(t) = t, \tag{3.19}$$

$$\dot{\bar{T}}_s^{\alpha}(t) \geq 0, \forall s \in \mathcal{S}, \alpha \in \mathcal{C}(s). \tag{3.20}$$

$$\dot{\bar{Y}}_i(t) \geq 0, \dot{\bar{Z}}_j(t) \geq 0, \dot{\bar{F}}_j(t) \geq 0, \forall i \in \mathcal{V}, j \in \mathcal{J}, t > 0. \tag{3.21}$$

The new continuous system is essentially the same as compared to the original discrete stochastic system, except that all processes are now *deterministic*. Thus, the fluid limits allow much easier analysis than the original stochastic system. Further, the power of fluid limits that, the stability guarantees in the continuous system can be extended to the original system, due to the following lemma [74]:

Lemma 3.1.1. *Suppose $\bar{U}_i(t) = 0$ for any $t > 0$ if $\bar{U}_i(0) = 0$ for any fluid limit. Then, the queue $U_i(n)$ is rate stable in the original stochastic queueing system.*

Proof: The proof is in Appendix B.2. ■

Thus, rate stability for a queue in the original stochastic system can be guaranteed by showing that the corresponding fluid queue is always zero if the initial queue length is zero. We now use this lemma to prove Theorem 3.1.1. Before that, we need to explore certain important properties of the queueing system in the fluid limits, and prove several technical lemmas.

We prove that the max-weight property in the original system according to Algorithm 3.1.1 and Algorithm 3.1.2 can be naturally extended to the fluid limits:

Lemma 3.1.2. *The following is true for any fluid limit under both Algorithm 3.1.1 and Algorithm 3.1.2:*

$$\dot{\bar{T}}_s^{\alpha}(t) = 0 \text{ if } \alpha \notin \arg\max_{\alpha \in \mathcal{C}(s)} \sum_{i \in \mathcal{V}} \bar{U}_i(t)\alpha_i - \beta \sum_{j \in \mathcal{J}} \bar{\Phi}_j(t) f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}) \tag{3.22}$$

for any $s \in \mathcal{S}$ and $\alpha \in \mathcal{C}(s)$.

Proof: The proof is in Appendix B.3. ■

We next prove the following important lemma, which shows the stability result in fluid limits under the augmented max-weight algorithm.

Lemma 3.1.3. *For any fluid limit, if $\bar{U}_i(0) = 0$ and $\bar{\Phi}_j(0) = 0$ for all $i \in \mathcal{V}$ and $j \in \mathcal{J}$, we have $\bar{U}_i(t) = 0$ and $\bar{\Phi}_j(t) = 0$ for any $t \geq 0$ under the augmented max-weight scheduling algorithm.*

Proof: The proof is in Appendix B.4. ■

We are now ready to prove the main theorem of this section.

Proof of Theorem 3.1.1: The rate stability for job queues are guaranteed by Lemma 3.1.1 and Lemma 3.1.3. Thus, we only need to prove cost optimality results. Assume that the claim is not true. Then, we can find a subsequence $\{r_n\}$ such that

$$\lim_{n \to \infty} \frac{1}{r_n} \sum_{\tau=1}^{r_n} \sum_{j \in \mathcal{J}} f_j(\alpha_{\mathcal{N}_j}(\tau); s_{\mathcal{N}_j}(\tau)) > f^{\star} + \epsilon. \tag{3.23}$$

Since $\epsilon = \sum_{j \in \mathcal{J}} \epsilon_j$, there must exist $j \in \mathcal{J}$, a positive constant $\epsilon' > 0$ and a subsequence $\{r_{n_k}\}$, such that

$$\lim_{k \to \infty} \frac{1}{r_{n_k}} \sum_{\tau=1}^{r_{n_k}} f_j(\alpha_{\mathcal{N}_j}(\tau); s_{\mathcal{N}_j}(\tau)) \geq f_j^{\star} + \epsilon_j + \epsilon'. \tag{3.24}$$

Now, we can also find a further convergent subsequence, which converges to a fluid limit. In the limit, we have

$$\begin{aligned}
\bar{\Phi}_j(1) &\geq \bar{\Phi}_j(0) + f_j^{\star} + \epsilon_j + \epsilon' - \bar{F}_j(1) && (3.25)\\
&= \bar{\Phi}_j(0) + f_j^{\star} + \epsilon_j + \epsilon' - \hat{f}_j^{\star} && (3.26)\\
&\geq \bar{\Phi}_j(0) + \epsilon' && (3.27)\\
&\geq \epsilon', && (3.28)
\end{aligned}$$

which contradicts Lemma 3.1.3. Thus, we conclude that the cost optimality holds and therefore the theorem holds. ■

3.2 AUGMENTED MAX-WEIGHT SCHEDULING WITHOUT COST KNOWLEDGE

In this section, we solve the optimization problem SCH-C. We remind the reader that SCH-C does not require knowledge of the optimal scheduling costs, but requires the scheduling algorithm to achieve the optimal scheduling cost asymptotically. For SCH-C, we will propose another version of the augmented max-weight scheduling algorithm, which is motivated by the Lyapunov optimization framework in [80] in the context of communication networks. We will generalize the algorithm to the broader area of CPS, and prove optimality results.

3.2.1 Augmented Max-Weight Scheduling Algorithm

The algorithm is shown in Algorithm 3.2.1. Compared to Algorithm 3.1.1, a major difference is that the virtual cost queues are replaced by a constant, since we do not assume knowledge about the optimal scheduling cost. Thus, the scheduling algorithm in (3.29) always tries to achieve a tradeoff between the queue length weighted job departures and the instantaneous scheduling cost in each time slot. Further, in order to improve the delay performance, a 'place holder' ζ_i is introduced for each user i. In below, we will show that this algorithm still achieves optimal scheduling cost asymptotically.

The performance of the above scheduling algorithm will be compared against the following

Algorithm 3.2.1 Augmented Max-Weight Scheduling without Cost Knowledge

1: For each time slot n, compute $\alpha(n)$ by solving the following optimization:

$$\begin{aligned} \text{maximize}_{\{\alpha_i,\chi_i\}} \quad & \sum_{i\in\mathcal{V}}(U_i(n-1)+\zeta_i)\alpha_i - \beta\sum_{j\in\mathcal{J}} f_j(\alpha_{\mathcal{N}_j}(n); s_{\mathcal{N}_j}(n)) \\ \text{subject to} \quad & h_k(\alpha_{\mathcal{N}_k}, \chi_{\mathcal{N}_k}; s_{\mathcal{N}_k}(n)) = 0, \forall k \in \mathcal{F} \\ & \alpha_i \in \mathcal{A}_i(s_i(n)), \forall i \in \mathcal{V} \\ & \chi_i \in \mathcal{O}_i(s_i(n)), \forall i \in \mathcal{V} \end{aligned} \tag{3.29}$$

where $\{\zeta_i\}$ and β are properly chosen positive constants.

2: Update queues $U(n)$ according to (2.3).

N-slot look-ahead scheduling problem:

$$\text{SCH-N:} \quad \min_{\{\alpha_i(n),\chi_i(n)\}} \quad \frac{1}{N}\sum_{n=1}^{N}\sum_{j\in\mathcal{J}} f_j(\alpha_{\mathcal{N}_j}(n); s_{\mathcal{N}_j}(n)) \tag{3.30}$$

$$\text{subject to} \quad h_k(\alpha_{\mathcal{N}_k}(n), \chi_{\mathcal{N}_k}(n); s_{\mathcal{N}_k}(n)) = 0, \forall k, 1 \le n \le N \tag{3.31}$$

$$\alpha_i \in \mathcal{A}_i(s_i(n)) \; \forall i \in \mathcal{V}, 1 \le n \le N \tag{3.32}$$

$$\chi_i \in \mathcal{O}_i(s_i(n)), \forall i \in \mathcal{V}, 1 \le n \le N \tag{3.33}$$

$$\frac{1}{N}\sum_{n=1}^{N}\alpha_i(n) \ge \frac{1}{N}\sum_{n=1}^{N}\Lambda_i(n) + \epsilon, \forall i \in \mathcal{V}. \tag{3.34}$$

Essentially, SCH-N is a restriction of SCH-C to the finite time interval $[0, N]$, where the stability constraints on the queues are replaced by (3.34), which requires that the average departure rate should be larger than the average arrival rate by ϵ.

Now, we assume that time is divided into frames, where each frame has N time slots. Denote the optimal scheduling cost for the above problem SCH-N during the m-th frame as $f_m^\star$. We have the following theorem:

Theorem 3.2.1. *Algorithm 3.2.1 achieves the following asymptotic average scheduling cost:*

$$\begin{aligned} &\limsup_{M\to\infty}\frac{1}{MN}\sum_{n=1}^{MN}\sum_{j\in\mathcal{J}} f_j(\alpha_{\mathcal{N}_j}(n); s_{\mathcal{N}_j}(n)) \\ &\qquad \le \limsup_{M\to\infty}\frac{1}{M}\sum_{m=1}^{M} f_m^\star + \frac{B_1 + B_2 N + \sum_{i\in\mathcal{V}}\alpha_i^{\max}\zeta_i}{\beta}. \end{aligned} \tag{3.35}$$

Further, the queue lengths can be bounded as follows:

$$\limsup_{M\to\infty} \frac{1}{MN}\sum_{n=1}^{MN}\sum_{i\in\mathcal{V}} U_i(n) \leq \frac{B_1+B_2N}{\epsilon} + \frac{\beta}{\epsilon}\limsup_{M\to\infty}\frac{1}{M}\sum_{m=1}^{M} f_m^{\star} + \sum_{i\in\mathcal{V}}(\frac{\alpha_i^{\max}}{\epsilon}-1)\zeta_i, \tag{3.36}$$

where B_1 and B_2 are sufficiently large constants.

The above theorem shows that we can achieve the optimal scheduling cost asymptotically without prior knowledge about its value. Further, it demonstrates an interesting $O(1/\beta)$ versus $O(\beta)$ tradeoff between the scheduling cost and average queue length, in the sense that we can achieve a scheduling cost gap on the order of $O(1/\beta)$, according to (3.35), while incurring an upper bound on the average queue length on the order of $O(\beta)$, as shown in (3.36). Such tradeoff is also shown in [80] in the context of communication networks. Intuitively, this is due to the fact that Algorithm 3.2.1 always tries to achieve a balance between minimizing the queue lengths and minimizing the instantaneous scheduling cost, where the weight is specified by β, as shown in (3.29). Thus, large β implies higher weight on the scheduling cost and larger queue length, and vice versa. In CPS applications, β has to be chosen carefully based on the desired scheduling cost performance and the tolerance on the delay.

It is important to notice that the processes $\{s_i(n)\}$ and $\{\Lambda_i(n)\}$ can be arbitrary, which include other well-known models, such as Markov processes as a special case. In order to emphasize such result, we propose the following corollary:

Corollary 3.2.1. *Let $\{\tilde{\alpha}_i(n)\}$ be any sequence of control actions such that the problem SCH-N is feasible for any N-slot frame. The following scheduling cost result holds :*

$$\limsup_{M\to\infty}\frac{1}{MN}\sum_{n=1}^{MN}\sum_{j\in\mathcal{J}} f_j(\alpha_{\mathcal{N}_j}(n);s_{\mathcal{N}_j}(n)) \leq \limsup_{M\to\infty}\frac{1}{MN}\sum_{n=1}^{MN}\sum_{j\in\mathcal{J}} f_j(\tilde{\alpha}_{\mathcal{N}_j}(n);s_{\mathcal{N}_j}(n)) + \frac{B_1+B_2N+\sum_{i\in\mathcal{V}}\alpha_i^{\max}\zeta_i}{\beta}, \tag{3.37}$$

where $\{\alpha_i(n)\}$ are the control actions under Algorithm 3.2.1. Further, the following average queue

length result also holds:

$$\limsup_{M\to\infty}\frac{1}{MN}\sum_{n=1}^{MN}\sum_{i\in\mathcal{V}}U_i(n)\leq\frac{B_1+B_2N}{\epsilon}+\sum_{i\in\mathcal{V}}(\frac{\alpha_i^{\max}}{\epsilon}-1)\zeta_i + \frac{\beta}{\epsilon}\limsup_{M\to\infty}\frac{1}{MN}\sum_{n=1}^{MN}\sum_{j\in\mathcal{J}}f_j(\tilde{\alpha}_{\mathcal{N}_j}(n);s_{\mathcal{N}_j}(n)), \quad (3.38)$$

where B_1 and B_2 are sufficiently large constants.

Thus, compared to an arbitrary feasible sequence of control actions $\{\tilde{\alpha}_i(n)\}$, which can be computed by assuming certain models such as Markov processes, the control actions specified by Algorithm 3.2.1 achieve an arbitrarily close average scheduling cost, while ensuring a guaranteed upper bound on average queue lengths, where the $O(1/\beta)$ versus $O(\beta)$ cost-delay tradeoff is specified by the parameter β.

3.2.2 Optimality Proof

We use the Lyapunov drift analysis method by Neely [80] to prove the optimality. The key to the proof lies in analyzing the drift of a Lyapunov function $L(n)$, which is defined as follows:

$$L(n)=\frac{1}{2}\sum_{i\in\mathcal{V}}(U_i(n)+\zeta_i)^2+\beta\sum_{j\in\mathcal{J}}\sum_{\tau=1}^{n}f_j(\alpha_{\mathcal{N}_j}(\tau);s_{\mathcal{N}_j}(\tau)). \quad (3.39)$$

Define the T-slot drift of the Lyapunov function starting from time slot n as

$$\Delta_T L(n)=L(n+T)-L(n). \quad (3.40)$$

We first provide a bound on the drift of $L(n)$ under Algorithm 3.2.1 over a single frame.

Lemma 3.2.1. *Under Algorithm 3.2.1, the N-slot drift of $L(n)$ for each frame m can be bounded as*

$$\Delta_N L(n_m)\leq-\epsilon\sum_{i\in\mathcal{V}}\sum_{\tau=1}^{N}(U_i(n_m+\tau-1)+\zeta_i)+\beta Nf_m^{\star}+N\sum_{i\in\mathcal{V}}\alpha_i^{\max}\zeta_i+NB_1+N^2B_2,$$

where $n_m=(m-1)N$ and B_1,B_2 are sufficiently large constants.

Proof: The proof is in Appendix B.5. ■

We next extend the above analysis from one frame to multiple frames:

Lemma 3.2.2. *The drift of $L(n)$ over the first M frames satisfies the following:*

$$\begin{aligned}\Delta_{MN}L(0) \;\leq\; & -\epsilon\sum_{\tau=1}^{MN}\sum_{i\in\mathcal{V}}(U_i(\tau-1)+\zeta_i) \\ & +\beta N\sum_{m=1}^{M}f_m^\star + MN\sum_{i\in\mathcal{V}}\alpha_i^{\max}\zeta_i + MNB_1 + MN^2B_2. \qquad (3.41)\end{aligned}$$

Proof: This can be obtained directly by summing the bound in Lemma 3.2.1 over M consecutive frames. ■

We are now ready to prove Theorem 3.2.1.

Proof: According to the bound in Lemma 3.2.2, the average cost over M frames under Algorithm 3.2.1 can be bounded as follows:

$$\begin{aligned} & \frac{1}{MN}\sum_{j\in\mathcal{J}}\sum_{\tau=1}^{MN}f_j(\alpha_{\mathcal{N}_j}(\tau);s_{\mathcal{N}_j}(\tau)) && (3.42)\\ \overset{(a)}{\leq} \; & \frac{1}{\beta MN}L(MN) && (3.43)\\ = \; & \frac{L(0)+\Delta_{MN}L(0)}{\beta MN} && (3.44)\\ \overset{(b)}{\leq} \; & \frac{L(0)}{\beta MN}+\frac{B_1+B_2N+\sum_{i\in\mathcal{V}}\alpha_i^{\max}\zeta_i}{\beta}+\frac{1}{M}\sum_{m=1}^{M}f_m^\star, && (3.45)\end{aligned}$$

where (a) is due to the definition in (3.39), and (b) is because of Lemma 3.2.2. Thus, the cost optimality holds after taking $M\to\infty$.

We now to prove the queue length bound. From (3.41) one can easily see that

$$\begin{aligned}&\frac{1}{MN}\sum_{\tau=1}^{MN}\sum_{i\in\mathcal{V}}(U_i(\tau-1)+\zeta_i) \\ &\quad\leq \frac{L(0)-L(MN)}{\epsilon MN}+\frac{\beta}{M\epsilon}\sum_{m=1}^{M}f_m^\star+\frac{1}{\epsilon}\sum_{i\in\mathcal{V}}\alpha_i^{\max}\zeta_i+\frac{B_1+NB_2}{\epsilon}, \qquad (3.46)\end{aligned}$$

from which (3.36) follows from above by taking $M\to\infty$. ■

3.3 APPLICATION: COORDINATED CHARGING OF ELECTRIC VEHICLES

In this section, we apply the max-weight scheduling algorithms to the important case of coordinated EV charging in smart grids. We first focus on the throughput performance of the max-weight scheduling algorithm in CPS by assuming constant cost, and show that the max-weight scheduling algorithm can achieve high EV penetration level in large-scale power systems, while ensuring that the power system can operate in a secure and reliable manner. Then, we consider the cost-aware scheduling and show that the max-weight algorithm in Section 3.1 can achieve near-optimal minimum variance total load profile for overnight EV charging application.

3.3.1 Throughput Results

We next investigate the throughput performance of the max-weight algorithm, and show that it can achieve high EV penetration in large-scale power systems. We start with the simulation setup.

3.3.1.1 Simulation Setup

We simulate a residential EV charging scenario with the standard IEEE 13-bus test feeder [81], which corresponds to a real-world distribution system. The topology of the test feeder is shown in Fig. 3.2, where the colored (black and gray) nodes represent the buses associated with residential loads. In order to demonstrate the potential of EVs in integrating intermittent renewable energy sources, it is assumed that a wind generator is installed at bus 671, which is the gray node in Fig. 3.2. The wind generation pattern for the simulation period is shown in Fig. 3.3, which is obtained from a real-world data trace in a Pennsylvania wind farm [82]. The simulation considers an overnight charging scenario from 7pm to 5am in the next morning. It is assumed that all EVs are always plugged-in during the whole simulation period, and therefore are always available for charging. The non-EV residential load profile is specified by the real-world data trace from the SCE website [83]. The total non-EV load profile is shown in Fig. 3.4, where wind generation at bus 671 is treated as negative load. The load at each bus is obtained by scaling the SCE load profile proportionally

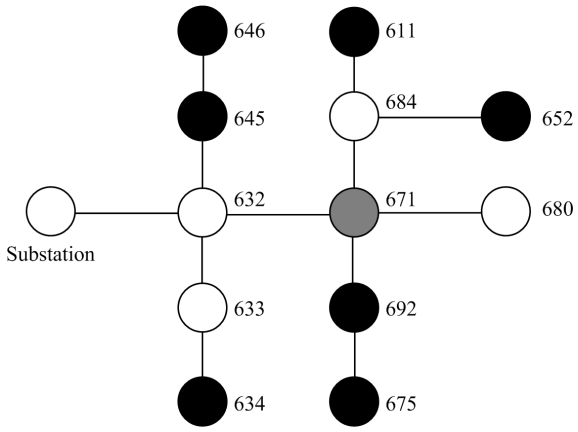


Figure 3.2: The topology of the standard IEEE 13-bus test feeder in the case study. The colored nodes are associated with residential loads. A wind generator is placed in the system at bus 671.

according to the case file description [81].

The EVs are assigned to the buses associated with residential loads, as shown by the colored nodes in Fig. 3.2. The number of EVs associated with each bus is proportional to the number of households for each bus, which is obtained according to the average daily load specification in the case file of the test feeder. For this simulation, the total number of EVs in the system is 2185, which corresponds to the 50% penetration scenario. It is assumed that the maximum charging power of each EV charger is 1.92kW, which corresponds to the standard 120V, 16A charger. For the charging simulation, it is assumed that the initial energy queue lengths for the overnight charging period for all EV batteries are 8.8 kWh. This is according to the national survey of 25 miles average daily commute distance, and the EV consumption rate of 34 kWh/100 miles [84]. A summary of the EV specification for this simulation is in Table 3.1.

3.3.1.2 Simulation Results

For this simulation, the optimal AC power flow in each time slot is computed by the technique of sequential convex programming [85], which works as follows. At each step, the algorithm tries to obtain a local convex approximation of the original nonconvex optimization problem, and then

Table 3.1: Vehicle Facts

Parameter	Value
Battery Capacity	16 kWh
Energy Usage per 100 miles	34 kWh
Charging Rate (120 V, 16 A)	1.92 kW
Average Daily Commute Distance	25 miles
Daily Consumption	8.75 kWh
Charging Efficiency	0.90

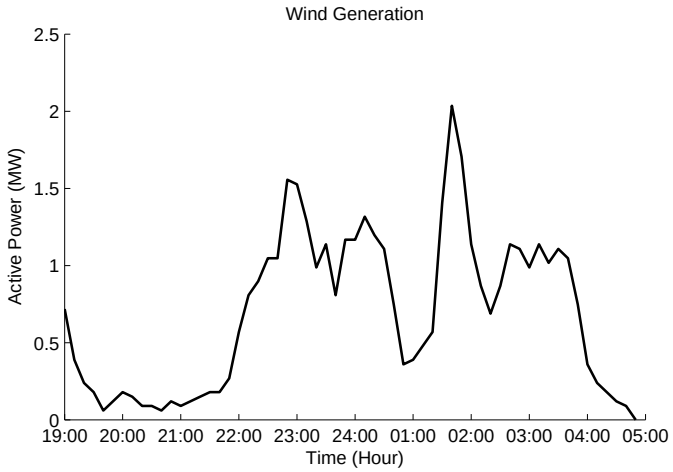


Figure 3.3: The wind generation output profile in the case study.

tries to solve the approximated convex problem in a local region and obtain the EV charging rates. The algorithm then solves the AC power flow with the updated EV charging rates, and continues to approximate the nonconvex power flow at the new operating point, and search for locally optimal solutions. The algorithm will stop if certain convergence criterion is satisfied. For this simulation, the AC power flow is solved using the OpenDSS software. The total computation time is around 10^3 seconds on a workstation with 64-bit Windows operating system running with 2.26GHz Intel Duo processor and 8GB RAM.

- *Total Load Profile*

  The resulting system load profiles are shown in Fig. 3.4, where the dotted line illustrates the non-EV load minus the wind generation, and the solid line corresponds to the total EV load.

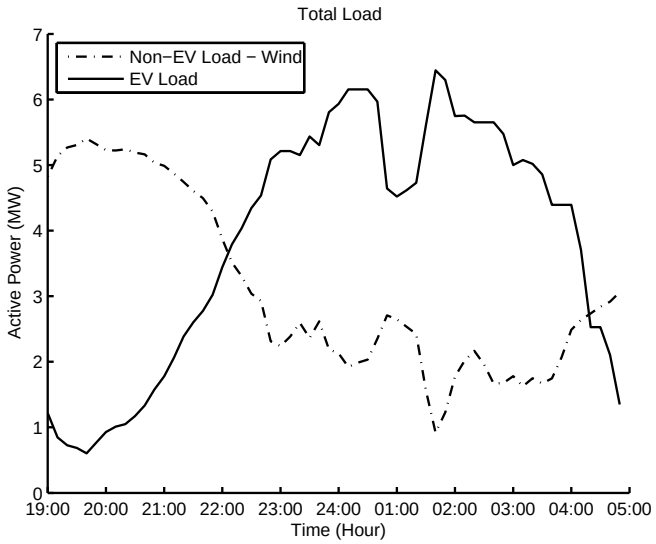


Figure 3.4: The load profiles according to the max-weight EV charging algorithm.

Note that the dotted load profile is no longer smooth, due to the integration of the highly intermittent wind generation. From the figure, one can clearly observe that the EV charging is 'smart', in the sense that the total EV load profile changes very adaptively to both the wind generation and non-EV load profiles. For example, during the peak hour (around 8pm), when the non-EV load is very large, the EV load is quite small, in order to guarantee that the physical limits are not violated, so that the power system can operate in a secure and reliable manner. Further, one can easily observe a 'symmetry' between the net load profile and the EV load profile, in particular during the mid-night, in that an increase in the dotted load profile usually results in a decrease in the EV load profile, and vice versa. In particular, as the dotted load suddenly drops around 2am, due to the sudden increase in the wind power generation output, one can clearly identify a very similar increase in the total EV charging profile. This immediately implies that the max-weight charging algorithm can successfully integrate the renewable wind generation by absorbing its intermittency. Finally, one can observe the sharp decrease in the total EV load in the morning of the next day. This indicates that most EVs are successfully refilled.

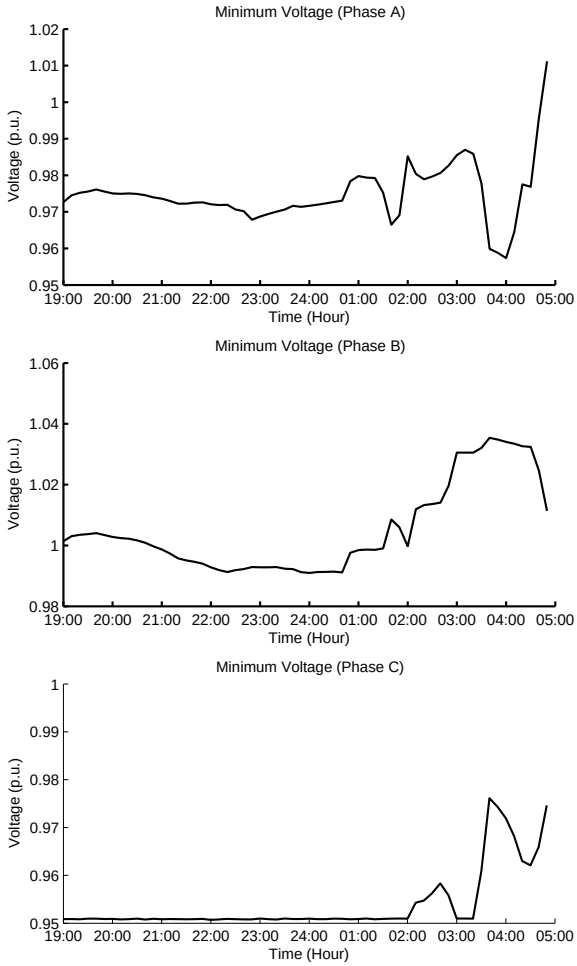


Figure 3.5: The profiles of the minimum three phase voltages in the case study.

- *Voltage Profile*

  The minimum voltage profiles for each phase in the case study are shown in Fig. 3.5. One can clearly observe that, phase C is the bottle neck of the system, as it has the smallest magnitude among all three phases. Note that, interestingly, even if the voltages in the other two phases are far away from the limit (0.95 per unit in this case study), the corresponding EV loads are still not allowed to charge more, due to the coupling between the phases. Further, note that the minimum voltage in the entire power system is always above the physical limit. Thus, we conclude that the max-weight charging algorithm can successfully control the charging

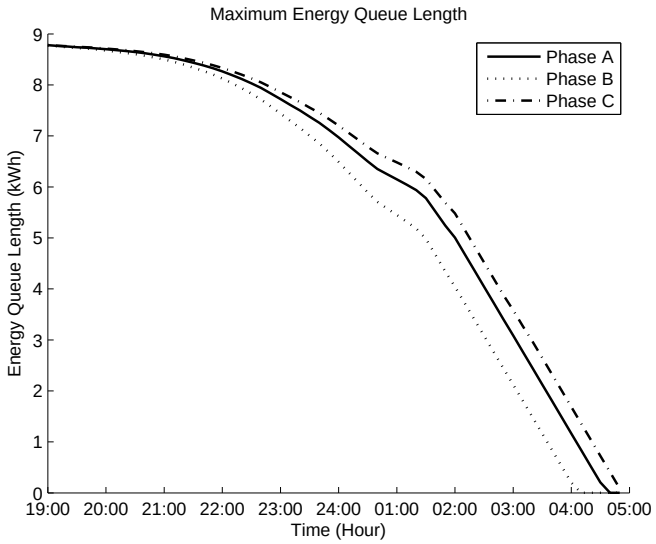


Figure 3.6: The profile of the maximum energy queue lengths for each phase in the case study.

rates of all EVs in the power system to maintain reliable operation of the power system. This also partially explains the symmetry between the dotted load profile and the EV load profile in Fig. 3.4, in that such constraint essentially places an upper bound on the total load in the power system, so that when the net load decreases due to wind power generation, the EV load will increase, and vice versa. Finally, one can observe the increase in the minimum voltage near the end of the overnight charging period. This is because many EVs finish charging.

- *Queue Length Results*

  In order to demonstrate the performance of the max-weight charging in refilling the EV batteries, we plot the profiles of the maximum energy queue lengths for each phase in Fig. 3.6. The conclusion is that, for all three phases, the max-weight charging algorithm can successfully refill all EV batteries during the overnight charging period. Further, the figure also confirms the coupling of the charging processes between the three phases, which is suggested in Fig. 3.5, in that even if the voltage limit in the phase A and B are far from the boundary, the EV loads are not allowed to charge further during the charging period, due to their coupling ef-

fect to the voltage in phase C, which is the bottleneck of the network. Thus, the maximum energy queue lengths in all three phases behave very similarly, with the EV loads in phase B finish relatively earlier, due to the fact that it is the least constrained in voltage, according to Fig. 3.5. Similarly, the EV loads in phase A also finish earlier than phase C. Further, a more careful inspection reveals that at the beginning of the charging period, the charging rate is relatively low, in order to avoid the power system congestion. The charging rate becomes much higher near the end of the charging period. This is because, during such period, the charging processes are essentially only constrained by the rating of the EV charging circuits.

3.3.2 Scheduling Cost Results

We next investigate the performance on scheduling cost on the augmented max-weight EV charging. Since the EV charging problem is a highly non-convex optimization problem, it is difficult to compute the optimal cost in general. Thus, in order to demonstrate the cost optimality, we assume that the physical voltage constraints can always be satisfied, and investigate the minimum variance EV charging problem for the overnight charging scenario. We assume that the cost for each time slot is as follows:

$$f(\{P_i(n)\};\{P_i^{\text{non-EV}}(n)\}) = \Big(\sum_{i\in\mathcal{V}}\big(P_i(n)+P_i^{\text{non-EV}}(n)\big)\Big)^2, \tag{3.47}$$

where $P_i(n)$ and $P_i^{\text{non-EV}}(n)$ are the EV charging power and non-EV active load at bus i, respectively. Thus, the optimal charging profile should be as flat as possible. We next describe the simulation setup.

3.3.2.1 Simulation Setup

The simulation setup is essentially the same as the one in the last subsection. We simulate the minimum variance charging in the standard IEEE 37-bus test feeder [81]. In this case, the total number of vehicles is 3402 for the 50% EV penetration scenario. For comparison purpose, there are

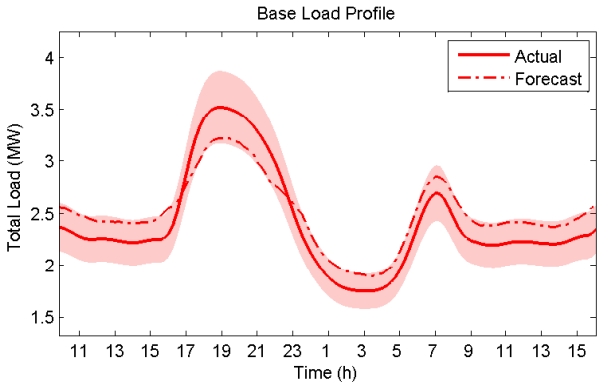


Figure 3.7: Base load profile used in the simulation with IEEE 37-bus system.

three types of smart charging algorithms considered in the simulation:

1. A static optimal minimum variance EV charging algorithm, with perfect knowledge of the day-ahead values of all random processes.

2. A static suboptimal charging algorithm, which solves the minimum variance EV charging using imperfect forecast of day-ahead load curve as shown in Fig. 3.7.

3. Augmented max-weight EV charging in Algorithm 3.2.1.

The charging algorithms are simulated at EV penetration levels of 30% and 50%. For the 30% penetration case, $\beta = 0.0205$, and $\zeta_i = 577$ for each vehicle, whereas for the 50% penetration case, $\beta = 0.0161$, and $\zeta_i = 534$ for each vehicle. The maximum total computation time of the on-line algorithm is 0.58 second for a 24-hour simulation scenario, while 3900 seconds for the static optimizations. Note the dramatic computation performance improvement for the case of max-weight charging. This is due to the fact that each charging schedule is computed using current system information, which have much smaller dimension than the total state processes. In practice, the time scale of each time slot is on the order of minutes. Thus, the computation and communication

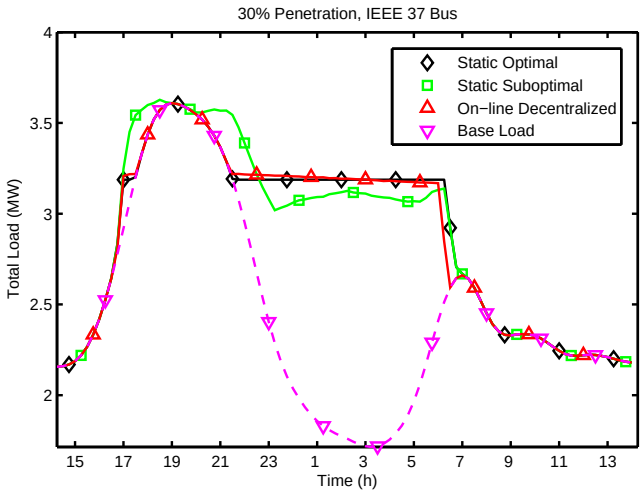


Figure 3.8: The total system load profile with 30% EV penetration in the IEEE 37-bus system.

requirement of the max-weight charging algorithm can be easily satisfied. The results of total loads are shown in Fig. 3.8 and Fig. 3.9, respectively. We have the following remarks.

- *Valley Filling*

  One can easily observe that the minimum load variance charging can achieve a *totally flat* load curve during the overnight charging period. Thus, compared to other smart charging formulations, in particular, the ones based on electricity price, the minimum load variance formulation can avoid an additional 'midnight peak', which, in the extreme case, may cause similar grid congestion issues as uncoordinated charging.

- *Cost Optimality*

  The proposed on-line decentralized EV charging achieves almost the same total load profile as the static optimal, *even though the former does not need to know the driving pattern and loads in advance*. This further verifies the theoretical result in Theorem 3.2.1 Thus, we can achieve the same performance as the static optimal, with much smaller computational overhead.

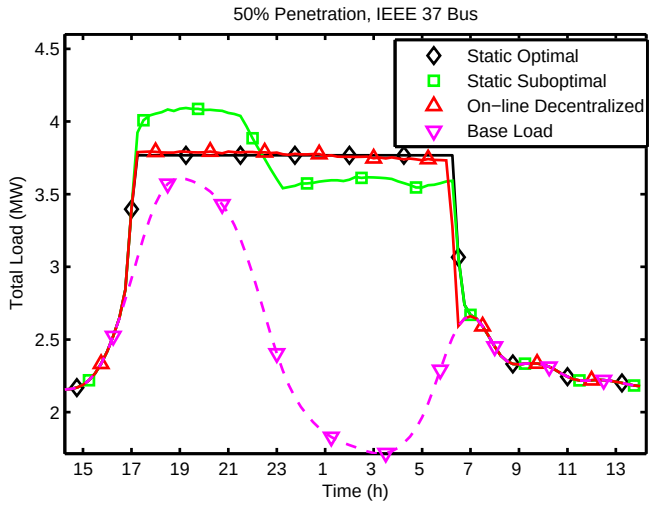


Figure 3.9: The total system load profile with 50% EV penetration in the IEEE 37-bus system.

- *Robustness Results*

  The day-ahead prediction based algorithms are vulnerable to the forecast errors. This can be clearly observed from Fig. 3.8 and Fig. 3.9, where the forecast based solutions cannot achieve a flat profile in the presence of the load forecast error. In fact, we allowed these algorithms to know the exact driving patterns in advance, which is clearly unrealistic. On the other hand, the optimal decentralized charging algorithm is not affected by such forecast errors, since it is an online algorithm, which does not rely on forecasts.

CHAPTER 4

OPTIMAL SCHEDULING IN THE QUASI-STATIC REGIME: SIMPLEX SCHEDULING

Chapter 3 discussed applications of the augmented max-weight scheduling algorithms in dynamic systems. This chapter considers the scheduling problem where the PhyNet operates in the *quasi-static* regime. That is, the local modes in the CPS remain constant for the time scale of the scheduling application. As one example, the data packets in a wireless sensor network are typically transmitted assuming a very slowly changing network topology. In such systems, it is possible to develop efficient scheduling algorithms by utilizing the quasi-static property of the system. In this chapter, we propose a simplex algorithm based optimal scheduling scheme applicable in the quasi-static regime, and prove its throughput optimality.

The main algorithm in this chapter is Algorithm 4.3.2, which proposes an optimal online simplex scheduling scheme. The algorithm has two components, the scheduling component and the column generation component. The scheduling component adopts a 'max-weight' form, in that a max-weight schedule is selected from a subset of 'basic' schedules. We will show that this is fundamentally different from the max-weight algorithms in Chapter 3, since the set of basic schedules has much smaller cardinality (e.g., $O(|\mathcal{V}|)$) than the set of all schedules (e.g., $2^{|\mathcal{V}|}$). Thus, the scheduling in Algorithm 4.3.2 is promising for distributed implementation, using average consensus techniques. Notice that this may incur higher complexity than the 'pick-and-compare' scheme in Al-

gorithm 3.1.2, since consensus has to be reached on the weights of all basic schedules, instead of the single newly generated schedule. Such increase in complexity achieves significant improvement on the steady-state delay, since Algorithm 4.3.2 will behave similarly to the conventional max-weight algorithm if the correct basic schedules are given. Finally, we will show that the column generation component in Algorithm 4.3.2 is also a max-weight problem, which can be similarly implemented in a distributed manner using the techniques discussed in Chapter 3.

We will apply simplex scheduling to packet scheduling in wireless networks. We will demonstrate that, by simulation results, the simplex algorithm can achieve dramatic steady-state delay improvement over the state-of-art CSMA based distributed scheduling algorithms [15, 16]. Further, we will also show that the simplex algorithm for packet scheduling in wireless networks can be implemented in a distributed manner, using average consensus and distributed CSMA techniques.

The rest of this chapter is organized as follows. In Section 4.1 we propose an idealized simplex scheduling algorithm, and in Section 4.2 we demonstrate the online scheduling algorithm and prove its optimality. Section 4.3 applies the simplex scheduling algorithm to packet scheduling in wireless networks, and show that it can be implemented in a distributed manner.

4.1 SIMPLEX SCHEDULING ALGORITHM: IDEALIZED VERSION

To motivate the development of the simplex-based scheduling, we need to reformulate the scheduling problem and system model.

4.1.1 A Reformulation of the Scheduling Problem

In this chapter, we are interested in solving the feasibility version SCH-F of the general scheduling problem formulations in Chapter 2. Since the system is in the quasi-static regime, the system mode is constant. Thus, we can enumerate all feasible schedules in a compact form as a matrix, which we denote as A, where each column $\alpha \in A$ represents a vector of feasible control actions. Now, the general abstract scheduling problem can be idealized by the following static linear pro-

gramming problem:

$$\text{SCH-L: minimize}_{\{x,\gamma\}} \quad \gamma \tag{4.1}$$

$$\text{subject to} \quad Ax = (1-\gamma)\lambda \tag{4.2}$$

$$x \succeq 0 \tag{4.3}$$

$$1^T x = 1 \tag{4.4}$$

where x is the scheduling variable, such that x_α represents the asymptotic time fraction that the control action vector α is chosen by the scheduler. Thus, the vector x naturally lives in the simplex as described in (4.3) and (4.4). (4.2) is essentially the rate stability constraint, where the LHS represents the average job departure rates, and the RHS represents the $(1-\gamma)$ discounted arrival rates, so that rate stability is achieved when the relaxation gap γ is non-positive. Given the above linear programming formulation, we next propose the simplex scheduling algorithm for the idealized problem SCH-L. We then prove its throughput optimality in the presence of stochastic job arrivals in the next section.

4.1.2 Idealized Simplex Scheduling Algorithm

Since the optimization SCH-L is a linear programming problem, we can solve it using the celebrated simplex algorithm [86]. The simplex based scheduling algorithm is shown in Algorithm 4.1.1. In order to fully understand the algorithm, we need to first introduce the concept of vertex in the context of the scheduling problem. (Note: 'vertex' in linear programming terminology is different from vertex in graph terminology.) According to the rate stability equality constraints in (4.2), we define a vertex as a pair $(y^T, \gamma)^T$, where y is a $|\mathcal{V}| \times 1$ sub-vector of x, which is associated with a $|\mathcal{V}| \times |\mathcal{V}|$ sub-matrix of A. Following the terminology in linear programming, we denote the sub-matrix as basic matrix B. We assume that the problem is *non-degenerate*, so that the matrix B is always invertible throughout the analysis in this chapter. Thus, the vertex can be obtained from

Algorithm 4.1.1 Static Simplex Scheduling

1: **Initialization:** Initialize the scheduling variables as the following:

$$B = \text{diag}(\alpha_1^{\max}, \alpha_2^{\max}, \ldots, \alpha_{|\mathcal{V}|}^{\max}) \tag{4.5}$$

$$y_i = \frac{\lambda_i/\alpha_i^{\max}}{\sum_{j\in\mathcal{V}} \lambda_j/\alpha_j^{\max}}, 1 \le i \le |\mathcal{V}| \tag{4.6}$$

$$\gamma = 1 - \frac{1}{\sum_{j\in\mathcal{V}} \lambda_j/\alpha_j^{\max}} \tag{4.7}$$

2: **if** $\gamma > 0$ **then**

3: **while** $\gamma > 0$ **do**

4: **Column Generation**: Compute a new column of A such that

$$\alpha_{\text{new}} = \arg \max_{\alpha \text{ is a column of } A} 1^T B^{-1} \alpha \tag{4.8}$$

5: **Scheduling**: Compute the new 'vertex' y_{new} and the throughput gap γ_{new} by solving the following optimization problem:

$$\begin{aligned} \text{minimize}_{\{y,z,\gamma\}} \quad & \gamma \\ \text{subject to} \quad & By + \alpha_{\text{new}} z = (1-\gamma)\lambda \\ & y \succeq 0, z \ge 0 \\ & 1^T y + z = 1 \end{aligned} \tag{4.9}$$

6: **Update**: Denote α' as the column in B whose coefficient in y_{new} is zero. Replace α' with α_{new}, relabel the variables in y_{new}, and update optimization variables as follows:

$$y = y_{\text{new}} \tag{4.10}$$

$$\gamma = \gamma_{\text{new}} \tag{4.11}$$

7: **end while**

8: **return** (B, y)

9: **end if**

the basis matrix B by solving the following:

$$\begin{pmatrix} B & \lambda \\ 1^T & 0 \end{pmatrix} \begin{pmatrix} y \\ \gamma \end{pmatrix} = \begin{pmatrix} \lambda \\ 1 \end{pmatrix} \tag{4.12}$$

Based on the above notion of a vertex, the static simplex scheduling algorithm works as follows. It starts from a feasible vertex, as shown in (4.6) and (4.7), which corresponds to the basis matrix B

in (4.5). Then, it generates a new moving direction α_{new} by solving (4.8), and obtain the new basic matrix and corresponding coefficients by solving (4.9). The above iteration continues until $\gamma \leq 0$, in which case the arrival rate can be fully stabilized by the basis matrix B. Denote $\mathcal{R}^\star$ as the convex hull of the columns in the matrix A. This is the largest stability region achievable by any scheduling algorithm [11]. We now prove that Algorithm 4.1.1 achieves $\mathcal{R}^\star$ in the following theorem.

Theorem 4.1.1. *If $\lambda \in \mathcal{R}^\star$, Algorithm 4.1.1 will return a solution (B, y) such that the following holds:*

$$By \succeq \lambda. \tag{4.13}$$

We first prove some technical lemmas. Firstly, we will show that the α_{new} returned in column generation step in (4.8) is a cost-decreasing direction.

Lemma 4.1.1. *After each iteration in Algorithm 4.1.1, the change to the scheduling cost function γ is non-positive, and is strictly negative if $\gamma > 0$.*

Proof: The proof is in Appendix C.1. ■

Given the new direction specified by α_{new}, according to the standard simplex algorithm, the coefficients should move along the direction as specified by $(\alpha_{\text{new}}^T, 1)^T$, until it reaches a new vertex, where some coordinate associated with one column of the old basis matrix B becomes zero for the first time. Then, Algorithm 4.1.1 replaces the column with α_{new}, and relabel the variables. We next prove the existence of such a column in Algorithm 4.1.1.

Lemma 4.1.2. *For the solution $(y_{new}^T, z_{new}, \gamma_{new})^T$ to the problem (4.9), we have $z_{new} > 0$, and there is one column in B whose corresponding coefficient in y_{new} is zero.*

Proof: The proof is in Appendix C.2. ■

We are now ready to prove the theorem.

Proof of Theorem 4.1.1: If γ in (4.7) is non-positive, we have

$$\sum_{j \in \mathcal{V}} \lambda_j / \alpha_j^{\max} \leq 1, \tag{4.14}$$

which implies that

$$By = \frac{1}{\sum_{j \in \mathcal{V}} \lambda_j / \alpha_j^{\max}} \lambda \succeq \lambda, \tag{4.15}$$

from which the claim holds. Thus, we only need to consider the case with $\gamma > 0$. In this case, Lemma 4.1.1 shows that each iteration moves along a cost decreasing direction. This, combined with the result in Lemma 4.1.2, implies that the algorithm moves to a new vertex after each iteration, which has a lower scheduling cost. Thus, the claim follows since the objective function is feasible, due to the assumption that $\lambda \in \mathcal{R}^\star$ and that there are a finite number of vertices for the feasible region. ■

Thus, we conclude that Algorithm 4.1.1 is optimal. Notice that one interesting property of the algorithm is that the scheduling phase is only restricted to a *sparse* set of schedules, which is represented by the basic matrix B. Such restriction can substantially simplify the computation and coordination overhead in each time slot, in particular compared to the augmented max-weight scheduling schemes in Chapter 3.

4.2 SIMPLEX SCHEDULING ALGORITHM: ONLINE VERSION

We have introduced the idealized simplex scheduling algorithm and proved its optimality in the last subsection. However, the algorithm design and analysis is still incomplete, due to the following. Firstly, the scheduling variables in Algorithm 4.1.1 are a static 'time fraction' result, which do not specify how the schedules are selected for each time slot. Thus, even if one finds the optimal basic scheduling variables $y^\star$, it is highly nontrivial to implement it efficiently in each time slot. Secondly, both algorithm design and optimality proof assume perfect knowledge of arrival rates. It is still unclear whether stability can be achieved by the simplex algorithm under very stochastic arrival processes with uncertain arrival rates. In this section, we continue with the simplex scheduling by proposing an online version and prove its optimality.

4.2.1 Scheduling Algorithm

For the online scheduling algorithm, we assume that time is partitioned into frames, where each frame has length T. At the beginning of each frame l, we estimate the arrival rates, as follows:

Estimate the arrival rates as follows:

$$\hat{\lambda}_i(l) = \epsilon_0 \lceil \frac{\Lambda_i\big((l-1)T\big)}{(l-1)T\epsilon_0} \rceil, \forall i \in \mathcal{V}, \tag{4.16}$$

where $\lceil \cdot \rceil$ is the standard ceiling function, and ϵ_0 is the quantization step size. Thus, the estimated arrival rate $\hat{\lambda}_i(l)$ is the quantized empirical arrival time-average arrival rates over the first $l-1$ frames, where the accuracy is specified by ϵ. Note that we always assume the 'rounding up' operation, in order to guarantee stability. The estimated arrival rate $\hat{\lambda}$ is then used by the scheduling algorithm throughout the entire frame.

The scheduling algorithm within each frame is shown in Algorithm 4.2.1. Note that the second step essentially refreshes the initial vertex in case there is a change in the arrival rates, so that the basic sets B is always feasible. Further, compared to the static version in Algorithm 4.1.1, there are a few major changes. Firstly, the 'scheduling' step in Algorithm 4.1.1 is replaced with the 'max-weight scheduling' in (4.21), where the parameter $\theta(n)$ is the dual variable for each rate stability constraint in (4.2). Secondly, the 'column generation' step in Algorithm 4.1.1 is replaced by another max-weight algorithm in (4.24). Notice the important difference between the two 'max-weight' algorithms. The first one in (4.21) searches over a much smaller set, namely the columns of B, whereas the second one search over the entire set of feasible schedules, the columns of A. The number of columns in B can be much smaller than that in A. For example, in wireless networks, the number of columns in A can be exponential in $|\mathcal{V}|$. Thus, the scheduling step in the online version is much easier to solve than the conventional max-weight algorithm [11]. Secondly, the online algorithm uses estimated arrival rates $\hat{\lambda}$ as shown in (4.16).

4.2.2 Stability Proof

We start the stability proof of the online simplex scheduling algorithm by showing that these changes are equivalent to the static versions in Algorithm 4.1.1. We begin with the following technical lemma.

Algorithm 4.2.1 Online Simplex Scheduling

1: Estimate arrival rate $\hat{\lambda}(l)$ with (4.16).
2: If $\hat{\lambda}_i(l) = \hat{\lambda}_i(l-1)$ for all $i \in \mathcal{V}$, the basis matrix B and scheduling variables (θ, γ) remain the same. Otherwise, initialize them as the following:

$$B = \text{diag}(\alpha_1^{\max}, \alpha_2^{\max}, \ldots, \alpha_{|\mathcal{V}|}^{\max}) \quad (4.17)$$
$$\gamma = 0 \quad (4.18)$$
$$\theta = 0 \quad (4.19)$$
$$\alpha_{\text{new}} = 0 \quad (4.20)$$

3: **for** $n = (l-1)T + 1 \to lT$ **do**
4: **Max-Weight Scheduling**: Choose $\alpha(n)$ such that

$$\alpha(n) \in \arg \max_{\alpha \text{ is a column of } B \text{ or } \alpha_{\text{new}}} \theta(n)^T \alpha \quad (4.21)$$

5: **Parameter Update**: The parameters are updated as follows:

$$\theta(n) = \theta(n-1) + \epsilon((1-\gamma(n-1))\hat{\lambda} - \alpha(n)) \quad (4.22)$$
$$\gamma(n) = \gamma(n-1) + \epsilon(\theta(n-1)^T \hat{\lambda} - 1) \quad (4.23)$$

where ϵ is a standard small constant step size.
6: **if** $(\theta(n), \gamma(n))$ converges and $\gamma > 0$ **then**
7: Replace the column in B with the minimum weight by α_{new}, and relabel coefficients.
8: Generate a new column α_{new} by solving the following:

$$\alpha_{\text{new}} \in \arg \max_{\alpha \text{ is a column of } A} \theta(n)^T \alpha \quad (4.24)$$

9: **end if**
10: **end for**

Lemma 4.2.1. *We assume that the estimated arrival rates $\{\hat{\lambda}_i\}$ is fixed, and that both the basic schedules B and α_{new} are fixed. Then, $(\theta(n), \gamma(n))$ will converge to the optimal primal and dual solutions for the optimization in (4.9), respectively.*

Proof: The proof is in Appendix C.3. ■

We continue to show that the second 'max-weight' algorithm in (4.24) is equivalent to the column generation step in (4.8) for the static optimization.

Lemma 4.2.2. *The new column α_{new} returned by the max-weight algorithm in (4.24) also solves the problem in (4.8).*

Proof: The proof is in Appendix C.4. ■

In the next lemma, we prove the result on average departure rates in steady states.

Lemma 4.2.3. *Assume that the estimated arrival rates $\{\hat{\lambda}_i\}$ is fixed at the quantized value of the true arrival rates, and that the throughput gap associated with the basic schedules B and α_{new} are non-positive. The following is true:*

$$\lim_{n\to\infty}\big\{(1-\frac{1}{2r_n\delta}\sum_{\tau=r_n(t_0-\delta)}^{r_n(t_0+\delta)}\gamma(\tau))\hat{\lambda}_i-\frac{1}{2r_n\delta}\sum_{\tau=r_n(t_0-\delta)}^{r_n(t_0+\delta)}\alpha_i(\tau)\big\}=0,\forall i\in\mathcal{V}, \tag{4.25}$$

for any $t_0>0$ and $\delta>0$.

Proof: The proof is in Appendix C.5. ■

We are now ready to prove the throughput optimality of the simplex scheduling algorithm.

Theorem 4.2.1. *Assume that the arrival rate $\lambda\in\mathcal{R}^\star$. The network is rate stable under the online simplex scheduling algorithm in Algorithm 4.2.1.*

Proof: Consider any fluid limit, and the following Lyapunov function:

$$L(t)=\frac{1}{2}\sum_{i\in\mathcal{V}}(\bar{U}_i(t))^2. \tag{4.26}$$

Let $t_0>0$ be given, we now show that

$$\dot{L}(t_0)=\sum_{i\in\mathcal{V}}\bar{U}_i(t_0)(\lambda_i-\dot{\bar{D}}_i(t_0))\le 0, \tag{4.27}$$

from which stability result holds after applying Lemma 3.1.1. Firstly, for any converging subsequence $\{r_{n_k}\}$ to the fluid limit, since we have

$$\lim_{k\to\infty}\sup_{t\in[0,T]}|\Lambda_i^{r_{n_k}}(t)-\lambda_i t|=0,\forall i, \tag{4.28}$$

for any $\epsilon'>0$, there exists K_1 such that

$$\sup_{t\in[0,T]}|\Lambda_i^{r_{n_k}}(t)-\lambda_i t|\le\epsilon',\forall i,k\ge K_1. \tag{4.29}$$

Now, we can choose ϵ' sufficiently small, such that (4.29) implies that the quantized estimated arrival rates in (4.16) stay unchanged at $\{\hat{\lambda}_i\}$, which is the quantized value of the true arrival rate. Note that we also have

$$\hat{\lambda}_i\ge\lambda_i,\forall i, \tag{4.30}$$

due to the 'round-up' quantization procedure in (4.16). Now, assume $t_0 > 0$ is given and that there is $i \in \mathcal{V}$ such that $\bar{U}_i(t_0) > 0$. Due to the uniform continuity property of the fluid limits and the uniform convergence on compact set, we can find $\delta > 0$, $\tilde{\epsilon} > 0$ and K_2 such that for $k \geq K_2$

$$U_i^{r_{n_k}}(\tau) \geq \tilde{\epsilon}, \forall \tau \in (t_0 - \delta, t_0 + \delta). \tag{4.31}$$

Recalling the definition of fluid scaling, this implies that

$$U_i(\tau) \geq r_{n_k}\tilde{\epsilon}, \forall \tau \in (r_{n_k}(t_0 - \delta), r_{n_k}(t_0 + \delta)). \tag{4.32}$$

Thus, for sufficiently large k, we conclude that U_i is always nonempty during $(r_{n_k}(t_0-\delta), r_{n_k}(t_0+\delta))$. Now, from Lemma 4.2.1 and Lemma 4.2.2, the Algorithm 4.2.1 is an implementation of the static version in Algorithm 4.1.1. Thus, for sufficiently large k, we conclude from Theorem 4.1.1 that the basic matrix B and α_{new} are such that the associated throughput gap γ is non-positive after $r_{n_k}t_0$. According to Lemma 4.2.3, we have

$$\lim_{k\to\infty} \Big\{(1 - \frac{1}{2r_{n_k}\delta} \sum_{\tau=r_{n_k}(t_0-\delta)}^{r_{n_k}(t_0+\delta)} \gamma(\tau))\hat{\lambda}_i - \frac{1}{2r_{n_k}\delta} \sum_{\tau=r_{n_k}(t_0-\delta)}^{r_{n_k}(t_0+\delta)} \alpha_i(\tau)\Big\} = 0, \tag{4.33}$$

from which and the convergence result of $\gamma(n)$, we conclude that

$$\begin{aligned} \lim_{k\to\infty} \frac{1}{2r_{n_k}\delta}\Big(D_i(r_{n_k}(t_0 + \delta)) - D_i(r_{n_k}(t_0 - \delta))\Big) &= (1 - \gamma)\hat{\lambda}_i & (4.34)\\ &\geq (1 - \gamma)\lambda_i. & (4.35) \end{aligned}$$

Taking $\delta \to 0$, we obtain that

$$\dot{\bar{D}}_i(t) \geq \lambda_i, \tag{4.36}$$

from which the stability holds. ■

Thus, we conclude that the online scheduling algorithm is optimal. We would like to emphasize the fundamental difference between the max-weight scheduling phase in (4.21) and the max-weight algorithms in Chapter 3, in that (4.21) is restricted to a very sparse set $(O(|\mathcal{V}|))$ of basic schedules, where as the algorithms in Chapter 3 always search over the entire set of feasible schedules. Thus,

Algorithm 4.3.1 Distributed CSMA

In each time slot n, do the following:
Randomly generate an independent set $\alpha'(n)$.
for each $i \in \alpha'(t)$ **do**
  $p_i = \exp(\theta_i)/(1+\exp(\theta_i))$;
  if no neighbor of i is in $\alpha'(n-1)$ **then**
    Link i update its transmission status as follows:

$$\alpha_i(n) = \begin{cases} 1 & \text{with probability } p_i \\ 0 & \text{else} \end{cases} \tag{4.37}$$

  end if
end for
Any other link $i \notin \alpha'(n)$ set $\alpha_i(n) = \alpha_i(n-1)$.

(4.21) has much lower complexity than the direct max-weight algorithm, and is amendable for distributed implementation.

4.3 APPLICATION: PACKET SCHEDULING IN WIRELESS NETWORKS

In this section, we will apply the simplex scheduling algorithm to the application of packet scheduling in wireless networks. In particular, we will demonstrate that the online simplex scheduling in Algorithm 4.2.1 can be implemented in a distributed fashion, using distributed CSMA [15, 16] and average consensus techniques.

4.3.1 Scheduling Algorithm

We start with the distributed CSMA algorithm, which can be regarded as a basic block in achieving distributed implementation of the max-weight column generation in (4.24).

The algorithm is shown in Algorithm 4.3.1. Notice that the carrier sensing is applied twice for each iteration of the algorithm. The first carrier sensing is applied during the generation of the independent set $\alpha'(n)$. We assume that $\alpha'(n)$ satisfies the following condition [16]:

$$\mathbb{P}(\alpha'(n) = \alpha) > 0, \forall \alpha \text{ feasible}. \tag{4.38}$$

The second carrier sensing is used to detect whether a neighbor of the link i is transmitting during time slot $n-1$. Thus, the algorithm is fully distributed, with no explicit message exchanges among links. The following lemma from [16] proves a product form stationary distribution of Algorithm 4.3.1.

Lemma 4.3.1. *The schedules $\{\alpha(n)\}$ in Algorithm 4.3.1 form a time-reversible Markov chain, with the following steady-state distribution:*

$$\pi_\alpha = \exp(\theta^T \alpha)/Z(\theta), \tag{4.39}$$

where $Z(\theta)$ is often referred to as the 'partition function':

$$Z(\theta) = \sum_{\alpha \text{ is a column of } A} \exp(\theta^T \alpha). \tag{4.40}$$

Thus, if we implement Algorithm 4.3.1 with parameter $\beta\theta$, where $\beta > 0$ is a large constant, from (4.39) we have

$$\pi_\alpha = \exp(\beta\theta^T \alpha)/Z(\beta\theta) \tag{4.41}$$

$$\approx \mathbf{1}_{\{\alpha \in \arg\max_{\tilde{\alpha} \text{ is a column of } A} \theta^T \tilde{\alpha}\}}, \tag{4.42}$$

which is an approximation of the max-weight schedule in (4.24). We will use this procedure as a building block to construct the distributed simplex scheduling.

The fully distributed scheduling algorithm is shown in 4.3.2. Compared to the online algorithm in Algorithm 4.2.1, the major differences are as follows:

- The first max-weight scheduling in (4.21) is implemented in a distributed manner with local weights updated by average consensus mechanisms.
- The second max-weight in (4.24) is implemented in a distributed manner by distributed CSMA.

It is important to notice that the first change is feasible because the number of columns in B is much smaller than the set of all feasible schedules, which may grow exponentially in the size of the network. Thus, the max-weight scheduling can be implemented using average consensus schemes with

Algorithm 4.3.2 Distributed Simplex Packet Scheduling

1: Estimate arrival rate $\hat{\lambda}(l)$ with (4.16).
2: If $\hat{\lambda}_i(l) = \hat{\lambda}_i(l-1)$ for all $i \in \mathcal{V}$, the basis matrix B and scheduling variables (θ, γ) remain the same. Otherwise, initialize them as the following:

$$B = I \tag{4.43}$$
$$\gamma = 0 \tag{4.44}$$
$$\theta = 0 \tag{4.45}$$
$$\alpha_{\text{new}} = 0 \tag{4.46}$$
$$\alpha_{\text{CSMA}} = 0 \tag{4.47}$$
$$\tag{4.48}$$

3: **for** $n = (l-1)T + 1 \to lT$ **do**
4: **Distributed CSMA**: Update $\alpha_{\text{CSMA}}(n)$ by running Algorithm 4.3.1 with large constant β.
5: **Distributed Max-Weight Scheduling**: Each link i computes

$$\alpha^{(i)}(n) \in \arg \max_{\alpha \text{ is a column of } B \text{ or } \alpha_{\text{new}}} w_{\alpha}^{(i)}(n) \tag{4.49}$$

where

$$w_{\alpha}(n) = \theta(n)^T \alpha \tag{4.50}$$

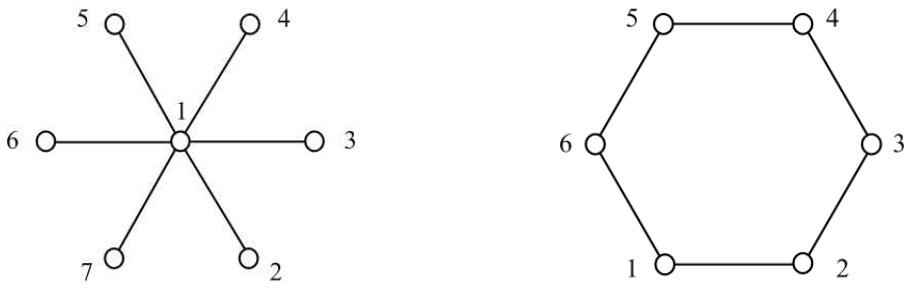
is the weight of independent set α, and $w_{\alpha}^{(i)}(n)$ is link i's local copy. Link i transmits if it has nonempty queue and $\alpha_i^{(i)}(n) = 1$.
6: **Parameter Update**: The parameters are updated as follows:

$$\theta(n) = \theta(n-1) + \epsilon\big((1-\gamma(n-1))\hat{\lambda} - \alpha(n)\big) \tag{4.51}$$
$$\gamma(n) = \gamma(n-1) + \epsilon\big(\theta(n-1)^T\hat{\lambda} - 1\big) \tag{4.52}$$

where ϵ is a standard small constant step size.
7: **Average Consensus:** Run an average consensus algorithm over the quantities $\{w_{\alpha}(n)\}_{\alpha \in B \cup \{\alpha_{\text{new}}\}}$ and $\gamma(n)$.
8: **if** $(\theta(n), \gamma(n))$ and $\alpha_{\text{CSMA}}(n)$ converges **then**
9: Replace the minimum weight column in B by α_{new}, and relabel coefficients accordingly.
10: Set $\alpha_{\text{new}} = \alpha_{\text{CSMA}}$.
11: **end if**
12: **end for**

low complexity, whereas the general max-weight scheduling problem is NP-hard. Summarizing the above discussions, we have the following theorem:

Theorem 4.3.1. *Let any feasible arrival rate $\lambda \in \mathcal{R}^{\star}$ be given. Assume that the average consensus in Algorithm 4.3.2 and the approximation in (4.42) are accurate. The network is rate stable under the fully distributed simplex scheduling algorithm in Algorithm 4.3.2.*



(a) (b)

Figure 4.1: (a) A star shaped interference graph for a wireless network with 7 links, and (b) A ring shaped interference graph for a wireless network with 6 links.

4.3.2 Simulation Results

In this subsection we demonstrate the performance of the distributed simplex packet scheduling in Algorithm 4.3.2 by simulation results. We will compare the performance of simplex scheduling against the hybrid queue-length-based distributed CSMA (HQ-CSMA) scheduling algorithm in [16], where the distributed CSMA scheduling in Algorithm 4.3.1 is applied to the links with large queue lengths (the threshold is chosen as 10^2). During the simulation, we assume that the packet arrivals are i.i.d with uniform arrival rates. The total simulation period is 3×10^5 time slots, and the initial queue length for each link is 10^3.

4.3.2.1 A Star Network

We first consider a star-shaped interference graph with 7 links in Fig 4.1 (a), with the simulation result shown in Fig. 4.2. In the figure, we plot the maximum queue length under the simplex scheduling and the queue lengths at link 1 and link 2 for the HQ-CSMA scheduling. Note that it is sufficient to focus on these two links, due to symmetry of the topology. We assume that the uniform arrival rate is at 95% of the capacity region boundary.

From the figure, one can clearly observe that the network is rate stable in both cases, and that HQ-CSMA scheduling has much larger queue lengths (around 10^3) than simplex scheduling (several

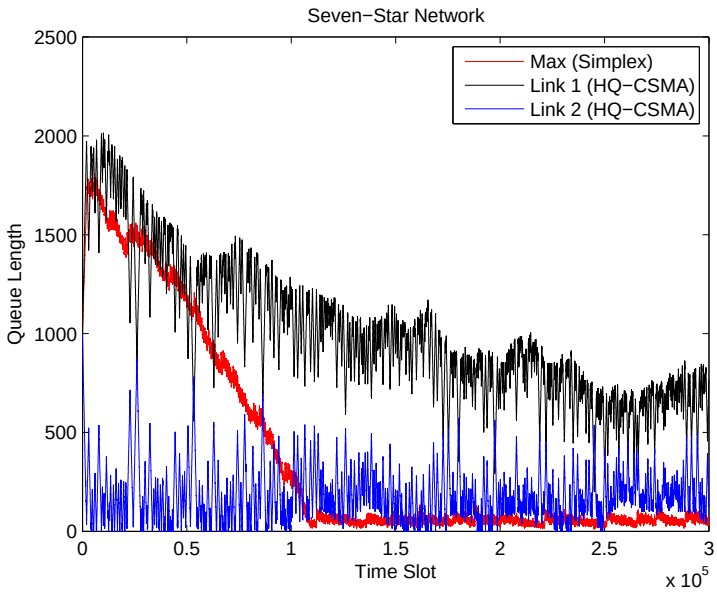


Figure 4.2: The simulation result of a 7-star network with HQ-CSMA scheduling and simplex scheduling.

hundreds) in the steady state. Further, one can observe that link 1 is the bottle neck for the HQ-CSMA scheduling, since its queue length is the largest almost all the time. This is because the HQ-CSMA scheduling spends a considerable amount of time around each 'good' schedule (such as the center link or the peripheral links) before transiting to the intermediate and suboptimal schedules. Notice that the HQ-CSMA achieves certain speed up by implementing the CSMA step only on the links with large queues, so that the center link 1 can quickly seize the channel when the queue lengths of all peripheral links are small. However, the transitions of schedules are still quite slow, due to the random-walk type design. On the other hand, simplex scheduling can quickly switch between the optimal basic schedules, and therefore, has much smaller queue lengths in steady states.

4.3.2.2 A Ring Network

We next consider a ring-shaped interference graph with 6 links in Fig 4.1 (b). The simulation result is shown in Fig. 4.3. Similar to the star network, we plot the maximum queue length under the simplex scheduling and the queue lengths at link 1 and link 2 for the HQ-CSMA scheduling.

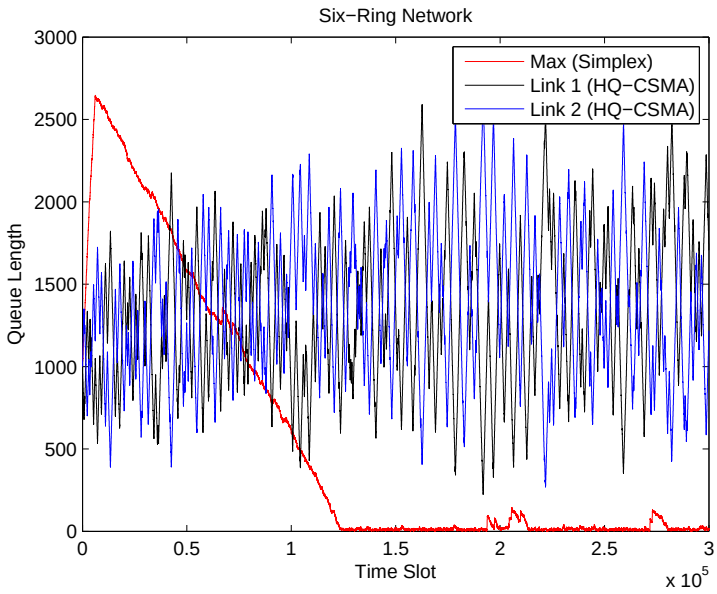


Figure 4.3: The simulation result of a 6-ring network with HQ-CSMA scheduling and simplex scheduling.

We assume that the uniform arrival rate is at 95% of the capacity region boundary.

One can easily observe that both algorithms can achieve rate stability. However, the simplex scheduling achieves much smaller queue lengths in steady states than the HQ-CSMA scheduling. This, again, demonstrates the fact that the simplex scheduler can achieve low delay by quickly switching between the optimal basic schedules. On the other hand, the switching between 'good' schedules for the HQ-CSMA scheme happens much less frequently, due to the random-walk type design. Further, one can observe that the HQ-CSMA is not achieving sufficient gain by restricting CSMA to the links with large queue lengths (greater than 10^2).

4.3.2.3 A Large Random Network

Finally, we consider the performance of the simplex scheduling algorithm in a large random network with 100 links, where the topology is shown in Fig. 4.4. The interference graph is constructed such that, two links form an edge if one's transmitter is within a certain distance from the receiver of the other link, where the threshold is computed assuming that the SINR threshold is 4.77dB, the

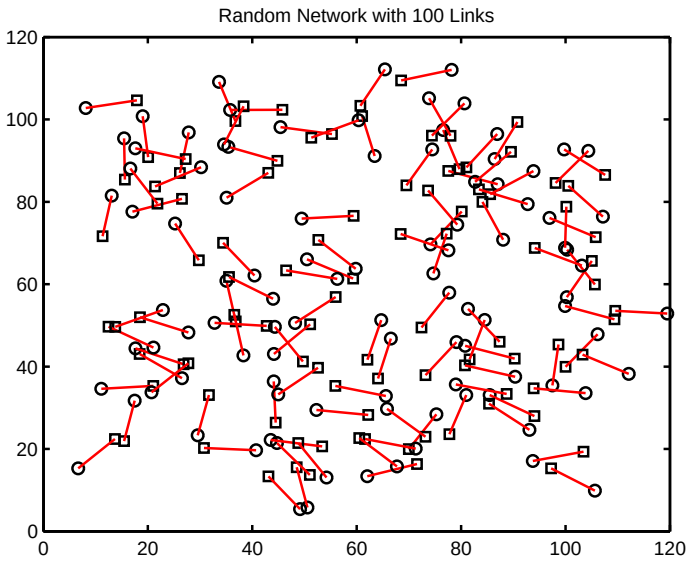


Figure 4.4: The topology of a large random network with 100 links.

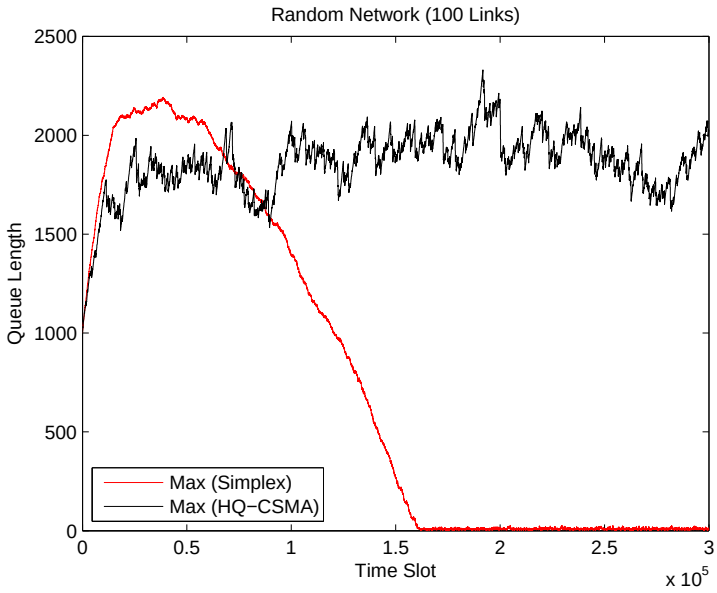


Figure 4.5: The simulation result of HQ-CSMA scheduling and simplex scheduling in a 100-link random network.

SNR is 20dB and the path loss exponent is 3. We assume that the uniform arrival rate is 0.1.

The simulation result is shown in Fig. 4.5, where we plot the maximum queue lengths for both

scheduling algorithms. One can easily observe that the network is rate stable under both scheduling algorithms, and that the simplex scheduling achieves much smaller queue lengths in steady states than the HQ-CSMA scheduling. Notice that the simplex algorithm may have larger queue lengths during the 'learning' period, since the algorithm needs to find all basic schedules. However, once all basic schedules are successfully computed, the queue lengths decreases dramatically, such that the delay performance is much better than the HQ-CSMA scheme, which needs sufficient amount of time to transmit between good schedules, due to the random walk design.

CHAPTER 5

SUBOPTIMAL SCHEDULING SCHEMES

The previous chapters have discussed optimal scheduling policies. Although optimal scheduling is desirable, such scheduling policies can be very difficult to implement in certain applications, due to the high complexity. For example, for the important case of packet scheduling in wireless networks, it is well known that optimal scheduling is NP-hard [87]. Thus, optimal scheduling schemes either incur exponential complexity in each time slot, such as the max-weight algorithm in [11], or incur exponential worst-case delay, such as the random 'pick-and-compare' algorithm in [12] and the distributed CSMA scheduling in [15, 16], where the exponential complexity is 'amortized' to achieve low scheduling complexity per time slot. Thus, suboptimal scheduling, even if it only achieves a fraction of the maximum throughput region, is still very attractive, due to the low complexity and ease of distributed implementation.

In this chapter, we investigate suboptimal scheduling policies for a restricted class of PhyNets. We are particularly interested in a class of low complexity scheduling policies, *maximal scheduling*. A maximal scheduler only specifies that the schedule in each time slot cannot be further augmented. Thus, compared to the max-weight scheduling schemes in Chapter 3 and simplex scheduling in Chapter 4, maximal scheduling is much simpler, since it only involves local user node interaction. Further, maximal scheduling is easily amendable for distributed implementation, such as using carrier sensing techniques for packet scheduling in wireless networks [66, 67]. For this reduction in

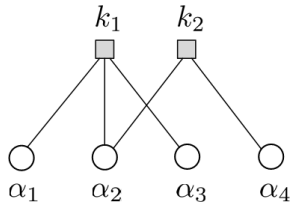


Figure 5.1: A simplified physical factor graph model for scheduling applications.

scheduling complexity is achieved at the expense of throughput region reduction, it is very important to provide throughput guarantees on the maximal scheduling schemes for general CPS applications.

In this chapter, we will investigate the throughput performance of maximal scheduling for the general scheduling problem in PhyNets. We focus on the feasibility formulation SCH-F in Chapter 2, and provide a lower bound on the stability region with an arbitrary maximal scheduling algorithm. We then show that it can achieve a certain fraction of the optimal stability region. We will also investigate specific maximal scheduling algorithms with improved throughput performances. In particular, we focus on static priority assisted maximal scheduling, and provide analysis for the application of packet scheduling in wireless networks. We will also show that the optimal static priority can be computed online with low complexity. Compared to conventional maximal scheduling, the static priority assisted maximal scheduling scheme can achieve dramatic throughput improvement.

The organization of this chapter is as follows. In Section 5.1 we introduce the simplified CPS system model, and in Section 5.2 we investigate the throughput performance of maximal scheduling with PhyNets. Section 5.3 discusses prioritized maximal scheduling. Finally, Section 5.4 discusses the application of maximal scheduling schemes to packet scheduling in wireless networks.

5.1 A SIMPLIFIED CPS SYSTEM MODEL

This chapter assumes a simplified system model of the general PhyNet model in Chapter 2. We assume that the system is quasi-static, so that the modes $\{s_i\}$ remain constant for the scheduling application. Since the physical variables $\{\chi_i\}$ are functions of the control variables $\{\alpha_i\}$, we eliminate them for the simplicity of discussion. We also assume that the physical factor nodes in (2.2) are linear. Thus, we can write the physical constraints in terms of the control actions $\{\alpha_i\}$ only. Thus, we have the following factor constraints:

$$\sum_{i \in \mathcal{N}_k} H_{ki}\alpha_i \leq 1, \forall k \in \mathcal{F}, \tag{5.1}$$

where $\{H_{ki}\}$ are coefficients as specified by the physical plant of the CPS. In this thesis, we assume that the coefficients H_{ki} are all nonnegative, so that the set of feasible schedules form an independence system, i.e., for any $\alpha' \preceq \alpha$, α is feasible implies that α' is also feasible. We are interested in the non-trivial cases and assume that

$$H_{ki}\alpha_i^{\max} < 1, \forall k, i \text{ such that } H_{ki} > 0. \tag{5.2}$$

Thus, each factor node $k \in \mathcal{F}$ involves at least two users, so that it represents network coupling. Intuitively, the user nodes in $\mathcal{N}_k$ form a *local conflict set* for *resource* k, in that their normalized weighted control actions cannot be larger than 1. An example factor graph model is shown in Fig. 5.1. In this case, assume that the feasible control actions are $\mathcal{A}_i = \{0, 1\}$ for each i and that $E_{ki} = 1/|\mathcal{N}_k|$ for all $i \in \mathcal{N}_k$. Thus, the following constraint has to be satisfied for any feasible α:

$$\frac{1}{3}\alpha_1 + \frac{1}{3}\alpha_2 + \frac{1}{3}\alpha_3 \leq 1, \tag{5.3}$$

$$\frac{1}{2}\alpha_2 + \frac{1}{2}\alpha_4 \leq 1. \tag{5.4}$$

The above model includes many important CPS applications as special cases. For example, it includes the hypergraph interference model in Section 2.4 as a special case. It can also be used for the problem of EV charging in power systems. For the EV charging application, (5.1) can be used

to model the constraint that the total load associated with a particular transmission line or transformer should be upper bounded, in a tree-structured distribution system. We next investigate the performance of maximal scheduling using the above physical network model.

5.2 MAXIMAL SCHEDULING

The maximal scheduling algorithm is very simple. According to the maximal scheduling criterion, the only requirement is that, in each time slot, the schedule is maximal, i.e., it cannot be further augmented. The scheduling is otherwise arbitrary. We say a schedule α' is an augmentation of α if $\alpha' \succeq \alpha$, and that at least one inequality is strictly satisfied. Thus, the only requirement on a maximal scheduler is that it has to generate 'maximal independent sets' of the independence system as described by (5.1). Thus, the scheduling algorithm has low complexity, and is promising to be implemented in distributed fashion. For example, for packet scheduling in wireless networks, a maximal scheduling algorithm can work as follows. In each time slot, the scheduler considers the back-logged links in an arbitrary manner, and adds a link i to the transmission schedule if there is no interference conflict when i is being considered. Fig. 5.2 shows an interference graph for wireless networks. In this case, if a maximal scheduler chooses the transmitting links according to the order $\{1, 2, 3, 4\}$, the resulting schedule is $\{1\}$. If the maximal scheduler choose transmitting links according to the order $\{4, 3, 2, 1\}$, the resulting maximal schedule is $\{4, 3\}$, which is also the maximum independent set of the interference graph.

Thus, compared to the optimal scheduling schemes in Chapter 3 and Chapter 4, maximal scheduling has low complexity, and is promising for distributed implementation in general CPS, since it only involves the local interactions of users. In this chapter, we analyze the throughput guarantees of maximal scheduling algorithms for the general scheduling problem in CPS. We first prove a lower bound on the stability region of maximal schedulers.

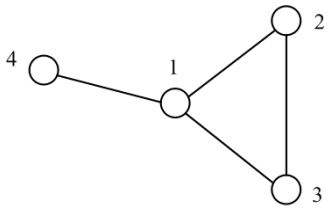


Figure 5.2: An example interference graph in wireless networks.

5.2.1 Stability Region

Before stating the stability guarantee, we need to introduce some notation first. Let the set $\mathcal{W}$ consists of all $|\mathcal{V}| \times |\mathcal{V}|$ matrices W which satisfy the following properties:

1. W is symmetric, and $W_{ij} \geq 0$ for all i and j.

2. $W_{ii} = 0$ for all i, and $W_{ij} = 0$ if $j \notin \mathcal{N}_i$, where the set $\mathcal{N}_i$ is the 'neighbor set' of user i, such that $j \in \mathcal{N}_i$ if and only if i and j are connected to a common factor node.

3. For any factor k that is connected to user i, we have

$$\sum_{j \in \mathcal{N}_k} W_{ij}\alpha_j \geq 1 \tag{5.5}$$

for any maximal schedule α satisfying $\alpha_i = 0$.

Intuitively, the matrix W assigns weights to job departures, such that the weighted departure for each active factor node k in a maximal schedule is larger than 1 when user i is idling, according to (5.5). We are now ready to state the following theorem on the lower bound stability region.

Theorem 5.2.1. *All queues in the system are stable for an arrival rate λ under any maximal scheduler if there is a matrix $W \in \mathcal{W}$, such that*

$$\frac{1}{\alpha_i^{\min}}\lambda_i + \sum_{j \in \mathcal{N}_i} W_{ij}\lambda_j \leq 1, \tag{5.6}$$

where $\alpha_i^{\min} > 0$ is the smallest positive value in $\mathcal{A}_i$:

$$\alpha_i^{\min} = \min_{\alpha_i \in \mathcal{A}_i, \alpha_i \neq 0} \alpha_i. \tag{5.7}$$

Proof: We only need to prove stability result in the fluid limit. That is, for any fluid limit, $\bar{U}_i(t) = 0$ for all $i \in \mathcal{V}$ and $t \geq 0$ if $\bar{U}_i(0) = 0$. Then, we can apply Lemma 3.1.1 to show stability in the original system. Let a fluid limit be given. Consider the following Lyapunov function:

$$L(t) = \frac{1}{2}\sum_{i\in\mathcal{V}} \bar{U}_i(t)\big(\frac{1}{\alpha_i^{\min}}\bar{U}_i(t) + \sum_{j\in\mathcal{N}_i} W_{ij}\bar{U}_j(t)\big). \tag{5.8}$$

We next calculate the derivative of $L(t)$ as follows:

$$\dot{L}(t) = \sum_{i\in\mathcal{V}} \frac{1}{\alpha_i^{\min}}\bar{U}_i(t)\dot{\bar{U}}_i(t) + \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}(W_{ij}+W_{ji})\bar{U}_i(t)\dot{\bar{U}}_j(t) \tag{5.9}$$

$$\overset{(a)}{=} \sum_{i\in\mathcal{V}} \bar{U}_i(t)\big(\frac{1}{\alpha_i^{\min}}\dot{\bar{U}}_i(t) + \sum_{j\in\mathcal{N}_i} W_{ij}\dot{\bar{U}}_j(t)\big) \tag{5.10}$$

$$\overset{(b)}{=} \sum_{i\in\mathcal{V}} \bar{U}_i(t)\big(\frac{1}{\alpha_i^{\min}}\lambda_i + \sum_{j\in\mathcal{N}_i} W_{ij}\lambda_j - (\frac{1}{\alpha_i^{\min}}\dot{\bar{D}}_i(t) + \sum_{j\in\mathcal{N}_i} W_{ij}\dot{\bar{D}}_j(t))\big), \tag{5.11}$$

where (a) is because the matrix W is symmetric, and (b) is because of SLLN. We only need to consider the case where there exists a user i and $t_0 > 0$ such that $\bar{U}_i(t_0) > 0$. In such a case, we will show that

$$\frac{1}{\alpha_i^{\min}}\lambda_i + \sum_{j\in\mathcal{N}_i} W_{ij}\lambda_j - \big(\frac{1}{\alpha_i^{\min}}\dot{\bar{D}}_i(t_0) + \sum_{j\in\mathcal{N}_i} W_{ij}\dot{\bar{D}}_j(t_0)\big) \leq 0, \tag{5.12}$$

from which one can conclude that $\dot{L}(t) \leq 0$, following which the theorem holds.

Now consider an arbitrary convergent subsequence $\{r_{n_k}\}_{k=1}^{\infty}$ associated with the fluid limit. Since $\bar{U}_i(t_0) > 0$, there is $\delta > 0$ such that

$$\bar{U}_i(t_0) > \delta > 0. \tag{5.13}$$

Further, since the function $\bar{U}_i(t)$ is uniformly continuous, there exists $\tau > 0$, such that

$$\bar{U}_i(t_0) > \frac{\delta}{2}, \forall t \in (t_0 - \tau, t_0 + \tau). \tag{5.14}$$

Thus, recalling the definition of fluid limit, for sufficiently large k we have

$$U_i^{r_{n_k}}(t) > \frac{\delta}{4}, \forall t \in (t_0 - \tau, t_0 + \tau), \tag{5.15}$$

which implies that

$$U_i(n) > \frac{r_{n_k}\delta}{4} \geq 1, \forall n \in (r_{n_k}(t_0-\tau), r_{n_k}(t_0+\tau)). \tag{5.16}$$

Thus, for sufficiently large k, user i always has nonempty queue during the time slots $(r_{n_k}(t_0-\tau), r_{n_k}(t_0+\tau))$. Finally, according to maximal scheduling, in each time slot, either user i has job departure, in which case $\alpha_i(n) \geq \alpha_i^{\min}$, or there is a factor node k which include user i, such that the corresponding constraint in (5.1) is active. In both cases, we have

$$\frac{1}{\alpha_i^{\min}}\alpha_i(n) + \sum_{j\in\mathcal{N}_i} W_{ij}\alpha_j(n) \geq 1, \forall n \in (r_{n_k}(t_0-\tau), r_{n_k}(t_0+\tau)), \tag{5.17}$$

due to the assumption in (5.5). Summing the above inequality over multiple time slots, we obtain the following:

$$\frac{1}{\alpha_i^{\min}}\big(D_i^{r_{n_k}}(t_0+\tau) - D_i^{r_{n_k}}(t_0-\tau)\big) + \sum_{j\in\mathcal{N}_i} W_{ij}\big(D_j^{r_{n_k}}(t_0+\tau) - D_j^{r_{n_k}}(t_0-\tau)\big) \geq 2\tau.$$

From which we conclude that, since τ can be arbitrarily small, in the fluid limit, we have

$$\frac{1}{\alpha_i^{\min}}\dot{\bar{D}}_i(t) + \sum_{j\in\mathcal{N}_i} W_{ij}\dot{\bar{D}}_j(t)) \geq 1. \tag{5.18}$$

Finally, we conclude from above and (5.6) that (5.12) holds. ■

Thus, we can achieve a guaranteed lower bound on the stability region for maximal scheduling. Notice that the important property is that the stability region is specified *locally*. This is because the scheduling algorithm only involves local interactions, so that the a user node i only needs to coordinate with the user nodes $\mathcal{N}_i$. Such local interactions simplifies the design of the scheduling algorithm. We next investigate the scheduling efficiency.

5.2.2 Scheduling Efficiency

As maximal scheduling is a class of suboptimal scheduling policies, we are interested in its performance compared to the optimal scheduling algorithm. Formally, this is defined by the scheduling

efficiency, as follows:

$$\gamma_\pi = \sup\{\rho \geq 0 : \rho\mathcal{R}^\star \subseteq \mathcal{R}_\pi\}, \tag{5.19}$$

where γ_π is the scheduling efficiency of scheduler π, $\mathcal{R}^\star$ is the optimal stability region, and $\mathcal{R}_\pi$ is the stability region associated with π. Thus, γ_π corresponds to the largest fraction of the optimal stability region $\mathcal{R}^\star$ that can be stabilized by π.

We need to make some definitions before stating the results about the scheduling efficiency for maximal scheduling. Define Δ_i associated each user i as follows. We first associate each neighbor $j \in \mathcal{N}_i$ with a weight Δ_{ij} as follows:

$$\Delta_{ij} = \frac{\mathbf{1}_{\{j\in\mathcal{N}_i\}}}{\min(\alpha_i^{\min}, \alpha_j^{\min})} \max(\nu_{ij}, \nu_{ji}), \tag{5.20}$$

where the term ν_{ij} is defined as follows

$$\nu_{ij} = \max_{\{k\in\mathcal{F}:\{i,j\}\subseteq\mathcal{N}_k\}} \max_{\alpha \text{ is maximal}, \alpha_i=0} \frac{1}{\sum_{j\in\mathcal{N}_k} \mathbf{1}_{\{\alpha_j>0\}}}. \tag{5.21}$$

We will show later that $\{\Delta_{ij}\} \in \mathcal{W}$. Note that $\Delta_{ij} = 0$ if i and j are not neighbors, and we define $\Delta_{ii} = 0$. Now, define Δ_i as follows:

$$\Delta_i = \max_{\alpha \text{ is maximal}} \Big\{ \frac{\alpha_i^{\max}}{\alpha_i^{\min}} \mathbf{1}_{\{\alpha_i>0\}} + \sum_{j\in\mathcal{N}_i} \Delta_{ij} \alpha_j^{\max} \mathbf{1}_{\{\alpha_j>0\}} \Big\}. \tag{5.22}$$

Intuitively, the above expression corresponds to an estimate of the total weight of job departures in each time slot in a neighborhood $\mathcal{N}_i$, where user i is associated with weight $\alpha_i^{\max}/\alpha_i^{\min}$, and user $j \in \mathcal{N}_i$ is associated with weight $\Delta_{ij}\alpha_j^{\max}$. Finally, define

$$\Delta = \max_{i\in\mathcal{V}} \Delta_i. \tag{5.23}$$

We will now show that $1/\Delta$ is a lower bound on the scheduling efficiency of maximal scheduling algorithms defined in (5.19).

Theorem 5.2.2. *The scheduling efficiency of any maximal scheduler π is bounded by*

$$\gamma_\pi \geq 1/\Delta. \tag{5.24}$$

Thus, if $\lambda \in \mathcal{R}^\star$, the network is stable under any maximal scheduler for any arrival process with average arrival rate λ/Δ.

We first present the outline of the proof. For any user i, in each time slot, we have

$$\begin{aligned}\frac{1}{\alpha_i^{\min}}\alpha_i(n)+\sum_{j\in\mathcal{N}_i}\Delta_{ij}\alpha_j(n) &\le \frac{\alpha_i^{\max}}{\alpha_i^{\min}}\mathbf{1}_{\{\alpha_i(n)>0\}}+\sum_{j\in\mathcal{N}_i}\Delta_{ij}\alpha_j^{\max}\mathbf{1}_{\{\alpha_j(n)>0\}}\\ &\le \Delta_i\\ &\le \Delta, \end{aligned} \tag{5.25}$$

according to the definition of Δ_i in (5.22). Thus, for any feasible arrival rate $\lambda \in \mathcal{R}^\star$, we have

$$\frac{1}{\alpha_i^{\min}}\lambda_i+\sum_{j\in\mathcal{N}_i}\Delta_{ij}\lambda_j \le \Delta. \tag{5.26}$$

Further, we will prove that the set of coefficients $\{\Delta_{ij}\} \in \mathcal{W}$, which implies that λ/Δ is in the lower bound region defined in Theorem 5.2.2.

In order to prove the theorem, we first need to prove two lemmas. We start with Lemma 5.2.1.

Lemma 5.2.1. *An arrival rate λ is stable under any maximal scheduler if*

$$\frac{1}{\alpha_i^{\min}}\lambda_i+\sum_{j\in\mathcal{N}_i}\Delta_{ij}\lambda_j \le 1,\ \forall i. \tag{5.27}$$

Proof: The proof is in Appendix D.1. ■

We next prove the following lemma, which proposes a necessary result on feasible arrival rates:

Lemma 5.2.2. *For any feasible arrival rate $\lambda \in \mathcal{R}^\star$, we have*

$$\frac{1}{\alpha_i^{\min}}\lambda_i+\sum_{j\in\mathcal{N}_i}\Delta_{ij}\lambda_j \le \Delta_i \le \Delta, \forall i. \tag{5.28}$$

Proof: The proof is in Appendix D.2. ■

Proof of Theorem 5.2.2: We can now prove Theorem 5.2.2. From the result in Lemma 5.2.2, we conclude that if $\lambda \in \mathcal{R}^\star$, then $(1/\Delta)\lambda$ must satisfy (5.27), and therefore, according to Lemma 5.2.1, is stable under any maximal scheduler π. ■

We have proved that $1/\Delta$ is a lower bound on the scheduling efficiency. Notice the interesting property that each Δ_i is defined locally. Thus, for many CPS applications with bounded neighborhood size $\max_{i\in\mathcal{V}}|\mathcal{N}_i|$, we can conclude that maximal scheduling can achieve a constant fraction

of the optimal stability region. Such property is very attractive in the systems where the optimal scheduling is hard to obtain.

5.3 PRIORITIZED MAXIMAL SCHEDULING

We have discussed the throughput guarantees of maximal scheduling and its scheduling efficiency. It should be noted that the class of maximal scheduling algorithms is very broad, due to its specification on 'arbitrary' maximal schedules. Thus, the worst case maximal scheduling may be quite suboptimal in certain cases. For example, it has been shown that [14] maximal scheduling in wireless networks under a 'unidirectional equal power' model may not achieve any positive fraction of the optimal stability region. In this section, we investigate performance improvements by designing specific maximal scheduling algorithms. We are particularly interested in *static priority assisted maximal scheduling schemes*, due to its simple design. Note that the maximal scheduler example for the wireless network in 5.2 at the beginning of the last section also serves as an example of static priority assisted maximal scheduling. For the general scheduling problem in CPS considered in this chapter, a static priority assisted maximal scheduler may work as follows. In each time slot, the scheduler will consider the back-logged users in a sequence specified by the static priority. When a user i is considered, it will choose the maximum feasible job departure rate, subject to the physical graph constraints in (5.1) and the constraint that its queue cannot be negative. It is easy to verify that the resulting schedule is maximal, since the set of schedules form an independence system.

Static priority assisted maximal scheduling is simple and easy to implement. Analysis of its throughput guarantees and the selection of the optimal priority, on the other hand, is very difficult. In this section, we provide throughput analysis of static priority assisted maximal scheduling and priority selection for wireless networks with interference graph constraints. The analysis and design for general CPS will be addressed in future research.

5.3.1 Maximal Scheduling with Static Priorities

We first introduce the concept of static priority. A priority vector p is defined as a permutation of $(1, 2, \ldots, |\mathcal{V}|)^T$, where p_i is the priority of link i. We say that link i has higher priority than link j if $p_i < p_j$. Thus, the link i with $p_i = 1$ has the highest priority, while the link j with $p_j = |\mathcal{V}|$ has the lowest priority. Given p, the prioritized maximal scheduler computes the schedule by considering the links sequentially, from the highest priority '1' to the lowest priority '$|\mathcal{V}|$', adding each back-logged link to the schedule if none of its higher priority neighbors have already been scheduled when it is considered. The following is a key property for the throughput guarantee of the scheduling scheme:

Lemma 5.3.1. *In any time slot, for any back-logged link i, a maximal scheduler with priority p will schedule at least one departure among the links $\{i\} \cup \mathcal{N}_i^p$, where $\mathcal{N}_i^p$ is the set of higher priority neighbors of link i.*

Proof: The proof is in Appendix D.3. ■

5.3.2 Stability Region

We next analyze the throughput performance of maximal scheduling assuming a fixed priority $\{p_i\}$ is always used. We first propose a lower bound stability region for maximal scheduling with static priority $\{p_i\}$.

Theorem 5.3.1. *The network is rate stable under maximal scheduling with static priority $\{p_i\}$ if the arrival rates satisfy the following:*

$$\mathcal{R}_p = \{\lambda \in \mathbb{R}_+^{|\mathcal{V}|} : \lambda_i + \sum_{j \in \mathcal{N}_i} \lambda_j 1_{\{p_i > p_j\}} \leq 1, \forall i \in \mathcal{V}\}, \tag{5.29}$$

where $1_{\{p_i > p_j\}}$ implies that only the neighbors with higher priority than link i are counted.

Essentially, the contribution of a priority in assisting a maximal scheduling algorithm is that it can reduce a neighborhood $\mathcal{N}_i$ to the 'higher priority neighborhood' in (5.29).

Proof: Since the priority $\{p_i\}$ is fixed, for ease of notation, we relabel the links in decreasing order of priorities according to $\{p_i\}$. Thus, link 1 has the highest priority, and link $|\mathcal{V}|$ has the lowest

priority. Consider the following Lyapunov function

$$L(t) = \frac{1}{2}\sum_{i\in\mathcal{V}} \bar{U}_i^2(t). \tag{5.30}$$

It is sufficient to prove that $\dot{L}(t) \leq 0$ if $\bar{U}_i(0) = 0$ for all $i \in \mathcal{V}$. Then, we can apply Lemma 3.1.1 to obtain stability in the original stochastic system. To prove this, in the following we will show that, by induction, $\frac{d}{dt}\bar{U}_i^2(t) \leq 0$ for each link i if $\bar{U}_i(0) = 0$ for all $i \in \mathcal{V}$.

We first consider the link 1, which has the highest priority according to $\{p_i\}$. Note that if $\bar{U}_1(t) = 0$, we have

$$\frac{1}{2}\frac{d}{dt}\bar{U}_1^2(t) = \bar{U}_1(t)\dot{\bar{U}}_1(t) \tag{5.31}$$

$$= 0. \tag{5.32}$$

Now suppose that, on the contrary, $\bar{U}_1(t) > 0$ at some $t > 0$. Then, there exists a constant $\epsilon > 0$ such that $\bar{U}_1(t) > \epsilon > 0$. Since $\bar{U}_1(t)$ is uniformly continuous, there also exists $\delta > 0$ such that

$$\bar{U}_1(\tau) > \epsilon/2, \forall \tau \in (t-\delta, t+\delta). \tag{5.33}$$

Now consider any converging subsequence $\{f^{r_{n_k}}(t)\}_{k=1}^{\infty}$ for the fluid limit. We have

$$U_1^{r_{n_k}}(\tau) > \epsilon/4, \forall \tau \in (t-\delta, t+\delta). \tag{5.34}$$

for sufficiently large k, which implies that

$$U_1(\tau) > r_{n_k}\epsilon/4 \geq 1, \forall \tau \in (r_{n_k}(t-\delta), r_{n_k}(t+\delta)). \tag{5.35}$$

That is, link 1 is always back-logged during the time interval $(r_{n_k}(t-\delta), r_{n_k}(t+\delta))$. Due to the prioritized maximal scheduling specification, link 1 transmits in every time slot in this interval, since it has the highest priority. Thus, we conclude that

$$D_1(r_{n_k}(t+\delta)) - D_1(r_{n_k}(t-\delta)) = 2r_{n_k}(t+\delta). \tag{5.36}$$

After taking limit as $k \to \infty$ we have

$$\bar{D}_1(t+\delta) - \bar{D}_1(t-\delta) = 2\delta, \tag{5.37}$$

which implies that $\dot{\bar{D}}_1(t) = 1$ since $\delta > 0$ can be arbitrarily small. Therefore, we conclude that

$$\begin{aligned}
\frac{d}{dt}\bar{U}_1^2(t) &= 2\bar{U}_1(t)\dot{\bar{U}}_1(t) && (5.38)\\
&= 2\bar{U}_1(t)(\lambda_i - \dot{\bar{D}}_1(t)) && (5.39)\\
&= 2\bar{U}_1(t)(\lambda_i - 1) && (5.40)\\
&\leq 0, && (5.41)
\end{aligned}$$

where the last equality is due to the assumption that $\lambda \in \mathcal{R}_p$. Thus, we have $\frac{d}{dt}\bar{U}_1^2(t) \leq 0$ and $\bar{U}_1(t) = 0$ for all $t \geq 0$.

We next proceed by induction. Suppose that $\frac{d}{dt}\bar{U}_k^2(t) \leq 0$ and $\bar{U}_k(t) = 0$ for all $t \geq 0$ and $k \leq l-1$, i.e., the first $l-1$ highest priority links. Now consider the link l, which has the l-th highest priority. Note that if $\bar{U}_l(t) = 0$ we have $\dot{\bar{U}}_l(t) = 0$. Now suppose $\bar{U}_l(t) > 0$ for some $t > 0$. Following the same argument as for link 1, we conclude that there is some interval $(r_{n_k}(t-\delta), r_{n_k}(t+\delta))$ during which $U_l(\tau)$ is nonempty. According to Lemma 5.3.1, in each time slot the maximal scheduler with priority $\{p_i\}$ will schedule at least one departure in $\{l\} \cup \mathcal{N}_l^p$, and therefore, we have

$$\begin{aligned}
&\Big((D_l(r_{n_k}(t+\delta)) + \sum_{j\in\mathcal{N}_l^p} D_j(r_{n_k}(t+\delta))\Big)\\
&\quad\geq \Big(D_l(r_{n_k}(t-\delta)) + \sum_{j\in\mathcal{N}_l^p} D_j(r_{n_k}(t-\delta))\Big) + 2r_{n_k}\delta
\end{aligned} \tag{5.42}$$

which implies, after taking $k \to \infty$, that

$$\dot{\bar{D}}_l(t) + \sum_{j\in S_l^p} \dot{\bar{D}}_j(t)\big) \geq 1. \tag{5.43}$$

Thus, we conclude that

$$\frac{d}{dt}\bar{U}_l^2(t) \stackrel{(a)}{=} 2\bar{U}_l(t)(\dot{\bar{U}}_l(t) + \sum_{j\in\mathcal{N}_l^p} \dot{\bar{U}}_j(t)) \tag{5.44}$$

$$= 2\bar{U}_l(t)(\lambda_l + \sum_{j\in\mathcal{N}_l^p} \lambda_j - \big(\dot{\bar{D}}_i(t) + \sum_{j\in\mathcal{N}_l^p} \dot{\bar{D}}_j(t)\big)) \tag{5.45}$$

$$\leq 2\bar{U}_l(t)(\lambda_l + \sum_{j\in\mathcal{N}_l^p} \lambda_j - 1) \tag{5.46}$$

$$\stackrel{(b)}{\leq} 0, \tag{5.47}$$

where (a) is because, by induction hypothesis, $\bar{U}_j(t) = 0$ for all $t \geq 0$ and all higher priority neighbors $j \in \mathcal{N}_l^p$, and (b) is because

$$\lambda_l + \sum_{j\in\mathcal{N}_l^p} \lambda_j \leq 1, \tag{5.48}$$

since $\lambda \in \Lambda_p$. Thus, by induction, we conclude that $\frac{d}{dt}\bar{U}_i^2(t) \leq 0$ for all $t \geq 0$ and all links in the network, from which the theorem follows. ■

Having proved that $\mathcal{R}_p$ is a lower bound stability region, we next show its tightness.

Theorem 5.3.2. *For any network, if $\mathcal{R}_p \neq \mathcal{R}^\star$, there exists an arrival rate vector $\lambda \in \mathcal{R}^\star$, which is arbitrarily close to $\mathcal{R}_p$, and a packet arrival process with average rate λ, such that the network is unstable under maximal scheduling with priority $\{p_i\}$.*

Proof: If $\mathcal{R}_p \neq \mathcal{R}^\star$, there must be an arrival rate $\lambda \in \mathcal{R}^\star$ such that for some link i, we have

$$\lambda_i + \sum_{j\in\mathcal{N}_i^p} \lambda_j > 1. \tag{5.49}$$

Further, the links in $\{i\} \cup \mathcal{N}_i^p$ can not form a clique, since in that case we will have $\lambda \notin \mathcal{R}^\star$. Thus, we can always find two independent links j and k in the set $\mathcal{N}_i^p$. Now consider the following arrival rates: $\lambda_i' = \epsilon$, $\lambda_j' = \lambda_k' = 1/2$, and $\lambda_l' = 0$ for any other link l. It is easily seen that $\lambda' \in \mathcal{R}^\star$, since one can simply alternate between the two schedules $\{i\}$ and $\{j,k\}$ in odd and even time slots to achieve network stability. Note that by adjusting the parameter ϵ, the arrival rate vector λ' can be arbitrarily close to $\mathcal{R}_p$. Now, we consider the following arrival process with arrival rate λ'. In every odd time slot, a packet arrives at link j, and in every even time slot, a packet arrives at link

k. Thus, according to the maximal scheduling with priority $\{p_i\}$, these packets are immediately transmitted in the next time slot. Finally, in each time slot, a packet arrives at link i independently with probability ϵ. Thus, link i is never scheduled by the maximal scheduler, and is therefore starved. ■

5.3.3 Scheduling Efficiency

We need to make some definitions before stating the results on scheduling efficiency. Given a fixed priority $\{p_i\}$, define Δ_i^p as the cardinality of the largest independent set in the subgraph induced by links $\{i\} \cup \mathcal{N}_i^p$. This is the set of transmitting links in the local neighborhood $\{i\} \cup \mathcal{N}_i^p$ with the maximum cardinality. We further define 'prioritized interference degree' Δ^p as

$$\Delta^p = \max_{i \in \mathcal{V}} \Delta_i^p. \tag{5.50}$$

We have the following theorem.

Theorem 5.3.3. *For any $\lambda \in \mathcal{R}^\star$, we have $(1/\Delta^p)\lambda \in \mathcal{R}_p$.*

Proof: For any link i, according to the definition of Δ_i^p, there are at most Δ_i^p packet departures among $\{i\} \cup \mathcal{N}_i^p$ in each time slot, since the transmitting links must form an independent set in the subgraph induced by $\{i\} \cup \mathcal{N}_i^p$. Thus, if the network is stable, the total average arrivals in $\{i\} \cup \mathcal{N}_i^p$ must be no more than the total average departures, i.e.,

$$\lambda_i + \sum_{j \in \mathcal{N}_i^p} \lambda_j \leq \Delta_i^p \leq \Delta^p, \forall i \in \mathcal{V}_I. \tag{5.51}$$

Multiplying both sides of the above inequality with $1/\Delta^p$, and recalling the definition of $\mathcal{R}_p$, we conclude that $(1/\Delta^p)\lambda \in \Lambda_p$ and the theorem follows. ■

Define $\mathcal{R}_{\text{sp}} = \cup_{p \in \mathcal{P}} \mathcal{R}_p$ as the union of the lower bound stability regions over all static priorities. This is the largest set of arrival rates that are guaranteed to be stable under all possible static priorities. Similarly, we can define $\Delta_{\text{sp}} = \max_{p \in \mathcal{P}} \Delta^p$. We will now show that $1/\Delta_{\text{sp}}$ is a lower bound on the scheduling efficiency of $\mathcal{R}_{\text{sp}}$.

Corollary 5.3.1. *For any $\lambda \in \mathcal{R}^\star$, we have $(1/\Delta_{sp})\lambda \in \mathcal{R}_{sp}$.*

Proof: Note that the set of priorities $\mathcal{P}$ is a finite set, and therefore there must exists $p^\star \in \mathcal{P}$, such that the following holds:

$$\Delta^{p^\star} = \Delta_{\text{sp}} = \min_{p \in \mathcal{P}} \Delta^p. \tag{5.52}$$

Thus, according to theorem 5.3.3, we have

$$(1/\Delta_{\text{sp}})\lambda = (1/\Delta^{p^\star})\lambda \in \mathcal{R}_{p^\star} \subseteq \mathcal{R}_{\text{sp}}, \tag{5.53}$$

from which the claim holds. ■

5.3.4 Optimal Priority Assignment

For the simplicity of exposition, we start with a simple offline scheme, where the priorities are computed with perfectly estimated packet arrival rates $\hat{\lambda}$. We will present a priority assignment and prove that it can produce a stabilizing priority as long as $\hat{\lambda} \in \mathcal{R}_{\text{sp}}$.

5.3.4.1 An Offline Assignment

The priority assignment algorithm is shown in Algorithm 5.3.1. At each step, the algorithm chooses a link k with the smallest 'total neighborhood arrival rate' $\hat{\lambda}_k + \sum_{j \in \mathcal{N}'_k} \hat{\lambda}_j$ in the *reduced* interference graph, and assigns it the lowest priority that is *locally* available. That is, link k only needs to have higher priority than the neighboring links which have already been removed. The algorithm then removes k from $\mathcal{V}'$ and repeats. We next show that Algorithm 5.3.1 implicitly solves the following min-max optimization problem:

Theorem 5.3.4. *The priority vector p returned by Algorithm 5.3.1 solves the following:*

$$p \in \arg\min_{p' \in \mathcal{P}} \max_{i \in \mathcal{V}} (\hat{\lambda}_i + \sum_{j \in \mathcal{N}_i^{p'}} \hat{\lambda}_j). \tag{5.55}$$

Proof: Let a priority $p' \in \mathcal{P}$ be given. It is sufficient to prove that

$$\hat{\lambda}_k + \sum_{j \in \mathcal{N}_k^p} \hat{\lambda}_j \leq \max_{i \in \mathcal{V}} (\hat{\lambda}_i + \sum_{j \in \mathcal{N}_i^{p'}} \hat{\lambda}_j) \tag{5.56}$$

Algorithm 5.3.1 Local Priority Assignment

1: **Initialize:** $\mathcal{V}' \leftarrow \mathcal{V}$;
2: **while** $\mathcal{V}'_I \neq \emptyset$ **do**
3: Choose link k such that

$$k = \arg\min_{i \in \mathcal{V}'} \{\hat{\lambda}_i + \sum_{j \in \mathcal{N}'_i} \hat{\lambda}_j\} \tag{5.54}$$

4: If no neighbor of link k has been removed, $p_k \leftarrow |\mathcal{V}|$. Otherwise $p_k \leftarrow \beta - 1$, where β is the lowest priority among the neighbors of link k which are already removed.
5: Removed link k from $\mathcal{V}'$ and its incident edges.
6: **end while**
7: **return** p

for any link $k \in \mathcal{V}$. For notation simplicity, we relabel the links according to the reverse order of the priority p, so that link 1 has the lowest priority, and link $|\mathcal{V}|$ has the highest priority. Now consider the first iteration of Algorithm 5.3.1, and denote $1'$ as the lowest priority link according to p'. We have

$$\hat{\lambda}_1 + \sum_{j \in \mathcal{N}'_1} \hat{\lambda}_j \overset{(a)}{\leq} \hat{\lambda}_{1'} + \sum_{j \in \mathcal{N}'_{1'}} \hat{\lambda}_j \tag{5.57}$$

$$\overset{(b)}{=} \hat{\lambda}_{1'} + \sum_{j \in \mathcal{N}^{p'}_{1'}} \hat{\lambda}_j \tag{5.58}$$

$$\leq \max_{i \in \mathcal{V}_I} (\hat{\lambda}_i + \sum_{j \in \mathcal{N}^{p'}_i} \hat{\lambda}_j). \tag{5.59}$$

Note that here, the sets $\mathcal{N}'_1$ and $\mathcal{N}'_{1'}$ refer to the neighbors of link 1 and $1'$ at the first iteration of Algorithm 5.3.1, respectively. (a) is because of (5.54), and (b) is because $\mathcal{N}'_{1'} = \mathcal{N}^{p'}_{1'}$, since link $1'$ has the lowest priority according to p'. Now consider the second iteration of Algorithm 5.3.1, with new reduced interference graph by removing link 1. Similarly, denote $2'$ as the lowest priority link according to p' in the reduced interference graph at the second iteration of Algorithm 5.3.1. We

have

$$\hat{\lambda}_2 + \sum_{j\in\mathcal{N}'_2} \hat{\lambda}_j \leq \hat{\lambda}_{2'} + \sum_{j\in\mathcal{N}'_{2'}} \hat{\lambda}_j \tag{5.60}$$

$$\overset{(a)}{\leq} \hat{\lambda}_{2'} + \sum_{j\in\mathcal{N}^{p'}_{2'}} \hat{\lambda}_j \tag{5.61}$$

$$\leq \max_{i\in\mathcal{V}_I}(\hat{\lambda}_i + \sum_{j\in\mathcal{N}^{p'}_i} \hat{\lambda}_j), \tag{5.62}$$

where (a) is because the set $\mathcal{N}^{p'}_{2'}$ refers to the original interference graph, which is a superset of $\mathcal{N}'_{2'}$, which is the set of higher priority neighbors in the reduced interference graph. Similarly, by repeating the above arguments, we conclude that

$$\hat{\lambda}_i + \sum_{j\in\mathcal{N}'_i} \hat{\lambda}_j \leq \max_{i\in\mathcal{V}_I}(\hat{\lambda}_i + \sum_{j\in\mathcal{N}^{p'}_i} \hat{\lambda}_j) \tag{5.63}$$

for each iteration of i of the Algorithm 5.3.1. Finally, according to Algorithm 5.3.1, the links removed later are always assigned higher priorities. Therefore, we have $\mathcal{N}^{p}_i = \mathcal{N}'_i$, which implies that

$$\hat{\lambda}_i + \sum_{j\in\mathcal{N}^{p}_i} \hat{\lambda}_j = \hat{\lambda}_i + \sum_{j\in\mathcal{N}'_i} \hat{\lambda}_j \tag{5.64}$$

$$\leq \max_{i\in\mathcal{V}_I}(\hat{\lambda}_i + \sum_{j\in\mathcal{N}^{p'}_i} \hat{\lambda}_j) \tag{5.65}$$

for all $i \in \mathcal{V}_I$, from which the theorem follows. ■

As an application of Theorem 5.3.4, we next prove that Algorithm 5.3.1 can achieve Λ_{sp}.

Theorem 5.3.5. *If $\hat{\lambda} \in \Lambda_{sp}$, Algorithm 5.3.1 will output a priority vector p such that $\hat{\lambda} \in \Lambda_p$.*

Proof: Since $\hat{\lambda} \in \Lambda_{\text{sp}}$, there is $p' \in \mathcal{P}$ such that $\hat{\lambda} \in \Lambda_{p'}$, which implies that

$$\max_{i\in\mathcal{V}_I}(\hat{\lambda}_i + \sum_{j\in\mathcal{N}^{p'}_i} \hat{\lambda}_j) \leq 1. \tag{5.66}$$

From Theorem 5.3.4, Algorithm 5.3.1 will return a priority p such that

$$\max_{i\in\mathcal{V}_I}(\hat{\lambda}_i + \sum_{j\in\mathcal{N}_i^p}\hat{\lambda}_j) \leq \max_{i\in\mathcal{V}_I}(\hat{\lambda}_i + \sum_{j\in\mathcal{N}_i^{p'}}\hat{\lambda}_j) \leq 1, \tag{5.67}$$

from which we conclude that $\hat{\lambda} \in \Lambda_p$. Therefore, the theorem follows. ■

5.3.4.2 Online Assignment

We next extend the offline version to the online case with estimated arrival rates from stochastic packet arrival processes, and prove that the same optimality result still holds. The online approach works as follows. We first partition time into frames, where each frame has duration of T time slots. A fixed priority $p(l)$ is used throughout an entire frame l. The computation of $p(l)$ is as follows. For the first frame, we assign $p(1)$ arbitrarily. At the beginning of each subsequent frame, we assign $p(l) = p(l-1)$ if the estimated arrival rate satisfies $\hat{\lambda}(l-1) \in \Lambda_{p(l-1)}$, where $\hat{\lambda}(l-1) = A((l-1)T)/(l-1)T$. Otherwise we set $p(l) = p$, where p is returned by Algorithm 5.3.1 with estimated arrival rates $\hat{\lambda}(l-1)$. We next show network stability in the following theorem:

Theorem 5.3.6. *The network is rate stable under the online priority assignment scheme if* $\lambda \in \mathbf{int}(\Lambda_{sp})$*, where* $\mathbf{int}(\cdot)$ *denotes the interior.*

Proof: We partition the set of priority vectors into three disjoint subsets:

$$\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{P}_3, \tag{5.68}$$

such that $\lambda \in \cap_{p\in\mathcal{P}_1}\mathbf{int}(\Lambda_p)$, $\lambda \in \cap_{p\in\mathcal{P}_2}\mathbf{bd}(\Lambda_p)$, and $\lambda \in \cap_{p\in\mathcal{P}_3}\Lambda_p^c$, where $\mathbf{int}(\cdot)$ denotes the interior, $\mathbf{bd}(\cdot)$ denotes the boundary, and $(\cdot)^c$ denotes the complement. Thus, λ is 'strictly' stable for any priority from $\mathcal{P}_1$, and is 'critically' stable for any priority from $\mathcal{P}_2$, but is unstable under any priority from $\mathcal{P}_3$. In the following, we will show that after a finite number of frames, the sequence of priority vectors $\{\hat{p}(l)\}$ will stay fixed at a priority vector in either $\mathcal{P}_1$ or $\mathcal{P}_2$. Thus, an identical argument using fluid limits as shown in the proof of Theorem 5.3.1 can be applied to show that the network is stable.

First, since $\lambda \in \cap_{p\in\mathcal{P}_1}\mathbf{int}(\Lambda_p)$, there exists an $\epsilon_1 > 0$ such that, for any $\hat{\lambda}$ satisfying $\|\hat{\lambda} - \lambda\|_2 <$

ϵ_1, we have $\hat{\lambda} \in \cap_{p\in\mathcal{P}_1}\mathbf{int}(\Lambda_p)$. Further, since λ is 'critically' stable under any priority in $\mathcal{P}_2$, we can choose $\epsilon_2 > 0$ such that for any $\hat{\lambda}$ satisfying $\|\hat{\lambda} - \lambda\|_2 < \epsilon_2$, and any $p \in \mathcal{P}_1$, $p' \in \mathcal{P}_2$, we have

$$\max_{i\in\mathcal{V}_I}(\hat{\lambda}_i + \sum_{j\in\mathcal{N}_i^p}\hat{\lambda}_j) < \max_{i\in\mathcal{V}_I}(\hat{\lambda}_i + \sum_{j\in\mathcal{N}_i^{p'}}\hat{\lambda}_j). \tag{5.69}$$

Thus, if Algorithm 5.3.1 is executed with the above $\hat{\lambda}$, the output priority vector must lie in $\mathcal{P}_1$, according to Theorem 5.3.4. Finally, note that $\cap_{p\in\mathcal{P}_3}\Lambda_p^c$ is an open set, we can choose $\epsilon_3 > 0$ sufficiently small, such that any $\hat{\lambda}$ satisfying $\|\hat{\lambda} - \lambda\|_2 < \epsilon_3$ still satisfies $\hat{\lambda} \in \cap_{p\in\mathcal{P}_3}\Lambda_p^c$. Now, we choose $\epsilon' = \min(\epsilon_1, \epsilon_2, \epsilon_3)$, and because of the SLLN, we can choose L to be large enough such that for any $l > L$, we have $\|\hat{\lambda}(l) - \lambda\|_2 < \epsilon'$. Thus, if Algorithm 5.3.1 is executed for any $l > L$, we have $p(l) \in \mathcal{P}_1$, because of (5.55). Further, for any $l > L$, if Algorithm 5.3.1 is executed, the priority vector will stay at the output result $p \in \mathcal{P}_1$, since by assumption, $\hat{\lambda}(l) \in \Lambda_p$. Finally, we only need to consider the case where Algorithm 5.3.1 is not executed for all $l \geq L$. It is clear that in such case, $p(l) \notin \mathcal{P}_3$ for any $l \geq L$. Thus, for sufficiently large l, the priority vector stays at a point in either $\mathcal{P}_1$ or $\mathcal{P}_2$ without invoking Algorithm 5.3.1, from which we can conclude that the network is stable. ■

5.4 APPLICATION: PACKET SCHEDULING IN WIRELESS NETWORKS

In this section, we apply the maximal scheduling algorithm schemes to the important application of the packet scheduling in wireless networks. We first apply the analysis in Section 5.2 to the wireless network scheduling with hypergraph interference model. Then, we will focus on the static priority assisted maximal scheduling, and demonstrate its performance by simulation.

5.4.1 Maximal Scheduling with Hypergraph Interference Model

As an application of the general maximal scheduling with PhyNets, we will show the throughput guarantees of maximal scheduling in wireless networks with general hypergraph interference

models. In below, we will investigate both stability region and scheduling efficiency, as a special case of the general results for PhyNets.

5.4.1.1 Stability Region

We first formulate the lower bound stability region. Similar to the definition for general PhyNets, let the set $\mathcal{W}$ consists of all $|\mathcal{V}| \times |\mathcal{V}|$ matrices which satisfy the following properties:

1. W is symmetric, and $0 \leq W_{ij} \leq 1$ for all i and j.

2. $W_{ii} = 0$ for all i, and $W_{ij} = 0$ if $j \notin \mathcal{N}_i$;

3. For any hyperedge e that includes link i, $\sum_{j \in e} W_{ij} \geq 1$.

We have the following theorem.

Theorem 5.4.1. *Let a maximal scheduler π with an interference hypergraph be given. Then, the network is stable under any arrival rate λ, if there is a matrix $W \in \mathcal{W}$, such that*

$$\lambda_i + \sum_{j \in \mathcal{N}_i} W_{ij} \lambda_j \leq 1, \forall i. \tag{5.70}$$

Note that if the hypergraph is indeed an interference graph, the matrix W is the graph incidence matrix: $W_{ii} = 0$, $W_{ij} = 1$ if $j \in \mathcal{N}_i$, otherwise $W_{ij} = 0$. Therefore, the above stability region reduces to the one proved in [14]. Thus, this lower bound region in Theorem 5.4.1 is a generalization of the lower bound for the graph model to the hypergraph models.

5.4.1.2 Scheduling Efficiency

Based on the above analysis on the stability region, we next investigate its scheduling efficiency. We first define the 'interference degree' Δ as follows. We first associate each neighboring link $j \in \mathcal{N}_i$ with a weight Δ_{ij} as follows:

$$\Delta_{ij} = \max_{e \in \mathcal{F}, \{i,j\} \subseteq e} \frac{1}{|e| - 1}, \tag{5.71}$$

where the hyperedge e has to include both links i and j ($\Delta_{ij} = 0$ if i and j are not neighbors). Now, define the interference degree of link i as follows:

$$\Delta_i = \max_{\alpha \text{ is a maximal schedule}} \alpha_i + \sum_{j \in \mathcal{N}_i} \Delta_{ij} \alpha_j. \tag{5.72}$$

In the graph case, this is equivalent to the maximum number of 'active edges', or simply the maximum number of concurrent transmissions in a link i's neighborhood [14], since $\Delta_{ij} = 1$ for all $j \in \mathcal{N}_i$. For general hypergraphs, we have $\Delta_{ij} < 1$, due to the fundamental property of cumulative interference. Finally, define $\Delta = \max_{i \in \mathcal{V}} \Delta_i$ as the interference degree of the hypergraph. As a special case of Theorem 5.2.2, we conclude that maximal scheduling with hypergraph interference models can achieve a scheduling efficiency of at least $1/\Delta$:

Theorem 5.4.2. *The queueing system is stable for any arrival process with arrival rate λ/Δ under any maximal scheduler π if $\lambda \in \mathcal{R}^\star$.*

We next discuss the tightness of the above lower bound on the scheduling efficiency. Note that if $\Delta = 1$, it is obvious that the scheduling efficiency is tight. We now assume that $\Delta > 1$, and show a tightness result in the following theorem:

Theorem 5.4.3. *Let a hypergraph be given, such that any link $i \in \mathcal{V}$ with $\Delta_i = \Delta > 1$ satisfies the following condition. The set of independent links in $\mathcal{N}_i$, which achieve an integer interference degree Δ, can be written as $\{e_1/\{i\}, e_2/\{i\}, \ldots, e_\Delta/\{i\}\}$, where the hyperedges $\{e_k\}$ are disjoint except a common link i. Then, for any $\epsilon > 0$, there is a feasible arrival rate $\boldsymbol{a} \in \mathcal{A}^\star$, and an arrival process with rate $\boldsymbol{a}'$, which is arbitrarily close to $\boldsymbol{a}$ in the sense that*

$$a'_j \leq (1/\Delta) a_j + \epsilon, \forall j \in \mathcal{V}. \tag{5.73}$$

Further, there is a maximal scheduler π such that the network is unstable under π with this arrival process.

Essentially, the theorem assumes that the hypergraph includes a generalized 'star' shaped hypergraph, where the independent set is a set of disjoint hyperedges (excluding link i).

Proof: Consider the following arrival rate vector λ: $\lambda_j = 1$ if and only if $j \in \{e_1, e_2, \ldots, e_\Delta\}/\{i\}$, otherwise $a_j = 0$. It is easily seen that $\lambda \in \mathcal{R}^\star$, since the set of links $\{e_1, e_2, \ldots, e_\Delta\}/\{i\}$ is an independent set. Now consider the arrival rate λ', such that $\lambda'_j = \lambda_j/\Delta$ if $j \neq i$, and

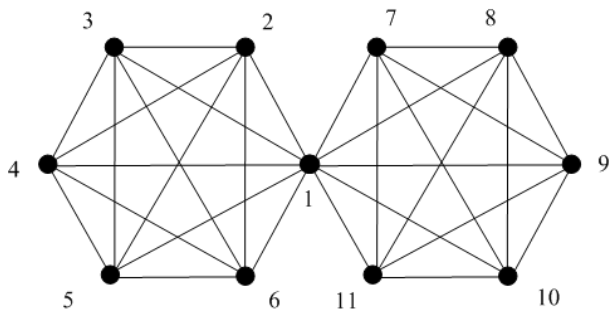


Figure 5.3: An interference graph of two cliques sharing one common link.

$\lambda'_i = \epsilon$. Thus, we have

$$\lambda'_j - (1/\Delta)\lambda_j = \epsilon \tag{5.74}$$

for all $j \in \mathcal{V}$. We next show that there exists an arrival process with such rate $\boldsymbol{a}'$, which makes the network unstable under a maximal scheduler π that assigns link i the lowest priority. That is, link i is always considered last by the scheduler π during scheduling. The arrival process is as follows. In each k-th time slots out of every Δ time slots, there is a packet arriving at each link in the set of links $e_k/\{i\}$. Then, it is immediately transmitted in the next time slot, because these links have higher priority than link i, and form an independent set. Further, it is easily seen that there is no departure from link i, since in each time slot, the transmitting links form an 'active' hyperedge with respect to link i. As far as link i is concerned, we assume that in each time slot, there is a packet arriving at link i with probability ϵ, so that $a'_i = \epsilon$. Thus, since link i never gets a chance to transmit, it is starved, and the network is unstable. ■

5.4.2 Prioritized Maximal Scheduling

In this section, we evaluate the performance of the proposed priority scheduling scheme by MATLAB simulation. All simulation results are obtained from 30 independent simulations over a period of 10^5 time slots. Three types of scheduling algorithms are mainly focused during simula-

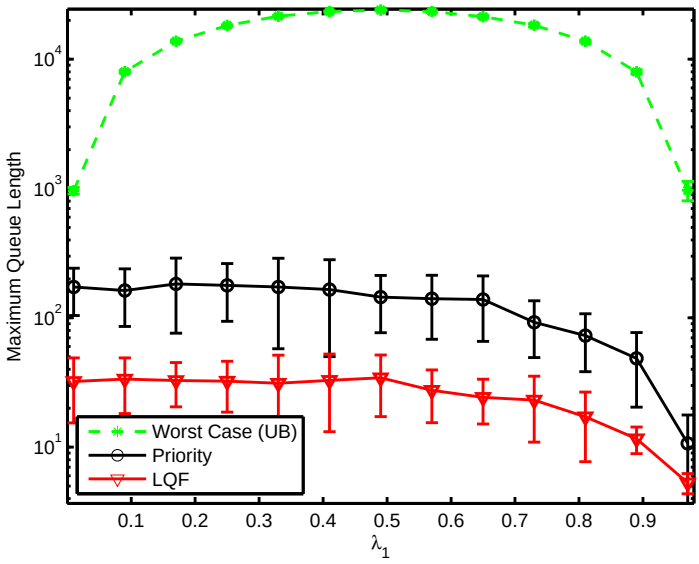


Figure 5.4: The performance of different scheduling schemes in the two-clique network .

tion: 1) a maximal scheduler with a suboptimal priority vector, as an upper bound on the worst-case throughput performance of maximal scheduling, 2) maximal scheduling with the online priority assignment algorithm, and 3) the LQF scheduling. Among these scheduling methods, only 2) requires estimation of arrival rates. For prioritized maximal scheduling, we choose $T = 100$.

5.4.2.1 Intersecting Cliques

We first consider a wireless network with 11 links as shown in Fig. 5.3, where the center link 1 is at the intersection of two cliques. Thus, link 1 interferes with both local clusters, and is the bottleneck of the network. We assume that every link other than link 1 has an arrival rate of $(0.99 - \lambda_1)/5$, so that each clique has a total arrival rate of 0.99. We further assume the arrival processes are independent Bernoulli processes. Thus, the online priority assignment algorithm converges very quickly. Fig. 5.4 shows the maximum queue lengths under different values of λ_1 with 95% confidence intervals.

- *Throughput Optimality*

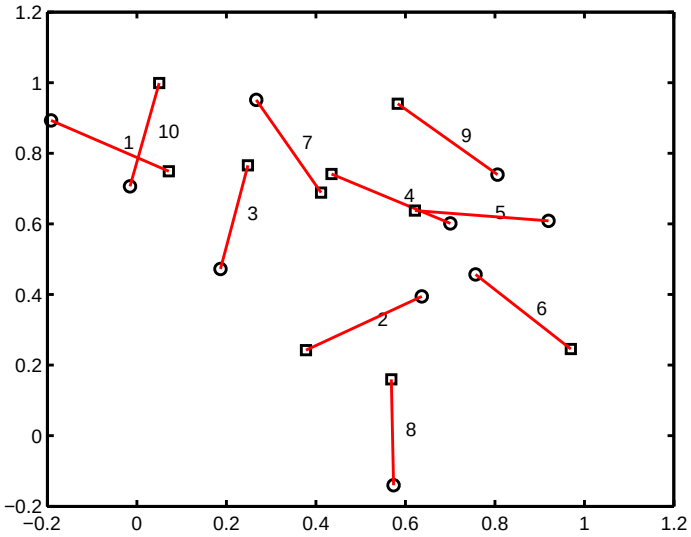


Figure 5.5: A random wireless network with 10 links. The square nodes are transmitters, and the round nodes are receivers.

The network is unstable under the worst-case maximal scheduling, which can be clearly observed by the very large queue lengths. On the other hand, the network is always stable under maximal scheduler with the optimal priority. In fact, for this topology, the optimal priority scheduling scheme is globally optimal, since one can easily verify that $\gamma_{\text{sp}} = 1$. Thus, we can obtain significant throughput improvement by properly optimizing the priorities.

- *LQF Scheduling*

  The network is stable under LQF scheduling. In fact, it can be shown that LQF scheduling is throughput optimal for such topology, due to the 'local pooling' condition [68]. In general, the LQF scheduling can achieve quite good throughput performance, at the expense of frequent update of global priorities. Compared to the LQF scheduling, the static priority based maximal scheduling can achieve similar throughput performance, with smaller scheduling overhead.

5.4.2.2 Random Topology

We next consider a random wireless network with 10 links, whose communication graph is shown in Fig. 5.5. To construct the interference graph, we place a guard zone [50] around the receiver of each link, so that two links form an edge if one's transmitter is inside the guard zone associated with the other. As a benchmark, we also simulate the optimal max-weight scheduling [11]. In order to demonstrate the convergence and sensitivity of the online priority assignment algorithm, we consider slowly converging arrival processes as shown in Fig. 5.6. All arrival processes have similar shape with different 'phases', and converge only after 10^4 time slots. Fig. 5.6 also shows priority updates at the corresponding links. One can clearly observe that our approach not only can quickly adapt to the empirical arrival rates in an online manner, but also is robust against the estimation errors, since the priorities change very infrequently with significantly oscillating empirical arrival rates. For this network, the maximum degree of the interference graph is 6, and the final priority assignment has 7 levels. Fig. 5.7 shows the maximum queue lengths after 10^5 time slots with 95% confidence intervals.

Remarks:

- *Throughput Optimality*

  Maximal scheduling with optimal priority achieves essentially the same maximum uniform throughput as the max-weight scheduling, although with larger queue lengths. This is in sharp contrast with the worst-case maximal scheduling, where the *ad hoc* choices of maximal schedules result in significant loss of throughput. One can easily observe that the maximal scheduling can only achieve a maximum throughput of 0.19, whereas the optimal priority achieves 0.25. Thus, we conclude that we can achieve significant throughput improvement by choosing the priority vectors carefully. Further, note that the max-weight scheduling has very high computational overhead. Thus, the optimal priority based maximal scheduling can achieve essentially the same throughput with much lower complexity.

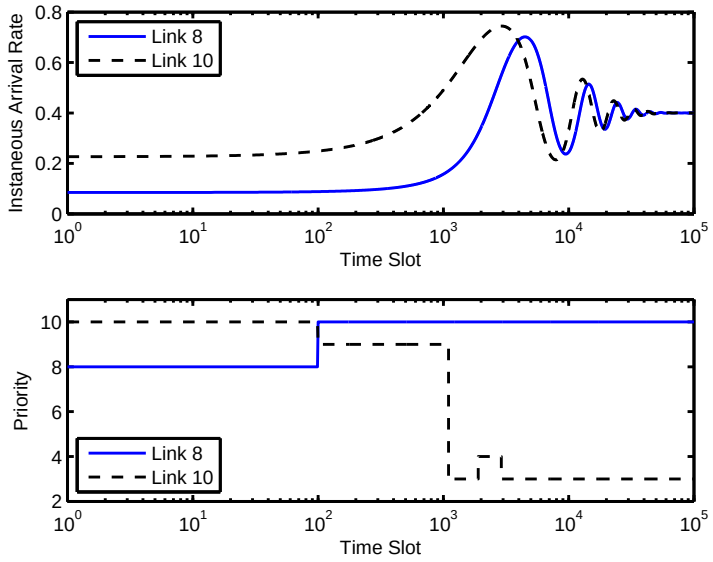


Figure 5.6: The top sub-figure shows the convergence of empirical arrival rates at link 8 and link 10, and the bottom sub-figure shows the convergence of their priorities. In the steady state, link 8 has the lowest priority '10', and link 10 has the highest priority '3'.

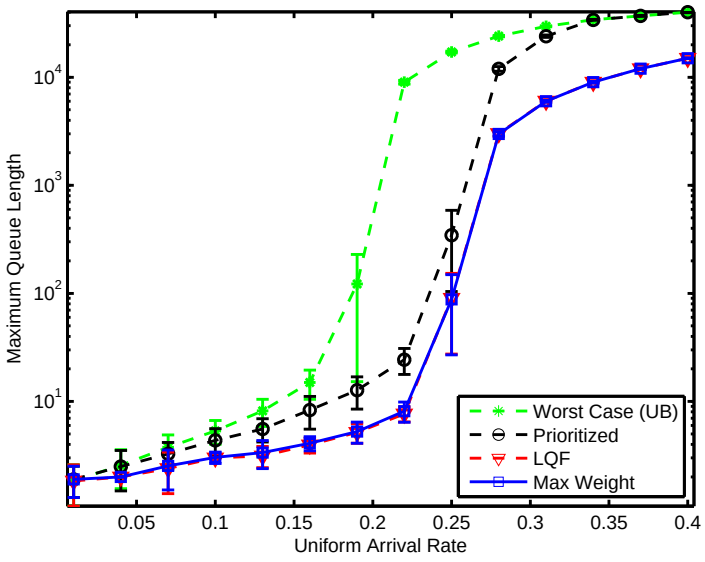


Figure 5.7: The simulation result in the random network with 8 links, where the maximum queue lengths are shown under uniform arrival rates.

- *LQF Scheduling*

  The LQF scheduling also achieves the network stability for all arrival rates, with smaller queue lengths than the optimal static priority. However, this is achieved at the expense of more priority computation overhead associated with changes in queue lengths. Note that it is possible to design similar multi-slot LQF (such as the T-slot updates in this paper) to further reduce the priority update overhead. However, LQF-type schemes typically incur larger overhead than our approach, since the queue lengths change more significantly than arrival rates in general. One can clearly observe this in Fig. 5.6, where the static priorities in the online approach change very infrequently. More in-depth investigation of the overhead and sensitivity issues will be addressed in future research.

CHAPTER 6

CONCLUSIONS

This thesis presented a general scheduling framework in physical networks, which covers a diverse range of important CPS applications. In the literature, such CPS applications were modeled and analyzed independently in the context of specific applications, such as packet scheduling in wireless networks, EV charging in smart grids, and workload scheduling in data centers. In this thesis, we showed that they can all be addressed in a unified manner, and we designed general scheduling schemes that can be applied to many applications. In this chapter, we provide a summary of the thesis and discuss future research directions.

6.1 SUMMARY

We started this thesis by proposing the general abstract scheduling problem in the context of PhyNets. We introduced the physical factor graph and the queueing system model, and formulated the general scheduling problem as a stochastic optimization problem. We then demonstrated broad applications of this general scheduling formulation to diverse research areas.

We then considered the design of optimal scheduling algorithms. We first focused on the category of dynamic regime, where the system modes in the CPS change randomly over time slots. In such case, we proposed augmented max-weight algorithms, which choose schedules myopically

in each time slot based on the current queue length information. We showed that, in the case with optimal cost knowledge, a virtual cost queue based max-weight algorithm can be used to achieve both asymptotic cost optimality and rate stability. We also proposed a 'pick-and-compare' version of the augmented max-weight algorithm, which has low complexity and is easy to be implemented in a distributed manner, using average consensus techniques. For the case without knowledge about optimal cost, a Lyapunov optimization based max-weight algorithm can also be used to achieve optimal cost asymptotically. Finally, augmented max-weight algorithms were investigated for the coordinated EV charging problem in power systems.

We next considered optimal scheduling in the quasi-static regime, where the system modes remain unchanged for the scheduling problem. In this case, it is possible to design more efficient scheduling algorithms by utilizing the quasi-static nature of the system. Inspired by the celebrated simplex algorithm, we proposed a simplex scheduling scheme, which chooses max-weight schedules among the set of 'basic' schedules. Since the set of basic schedules is 'sparse', the simplex scheduling can be implemented in a distributed manner using average consensus techniques. Further, we showed that the basic schedules can be solved by another max-weight problem. We proved the asymptotic throughput optimality of the simplex scheduling scheme with stochastic job arrivals. We finally applied the simplex algorithm to the important application of packet scheduling in wireless networks, and demonstrated that it can be implemented in a distributed fashion, using average consensus and distributed CSMA mechanisms. Simulation results showed significant steady-state delay reduction over the throughput-optimal distributed CSMA schemes.

Finally, we investigated the design and analysis of suboptimal scheduling algorithms. In this thesis, we focused on the class of maximal scheduling algorithms, which only require coordination of local user nodes, and therefore have low complexity and are easy for distributed implementation. We analyzed the throughput performance of maximal scheduling with PhyNets and proposed a lower bound on the stability region. We also showed that the maximal scheduling algorithm can achieve a certain fraction of the optimal throughput region. We then investigated the performance

improvement of maximal scheduling for packet scheduling in wireless networks, by utilizing static priorities. We analyzed the stability region associated with any fixed priority, and showed that the optimal static priority can be computed online with low complexity. We showed that the combined priority assignment and maximal scheduling approach achieve dramatic throughput improvement over conventional maximal scheduling algorithms.

6.2 FUTURE DIRECTIONS

We next point out several future research directions as a continuation of this thesis work. It should be emphasized that research on CPS is a huge and interdisciplinary topic, which covers many domains and a diverse range of applications. Thus, for a particular problem instance, it is important to adapt the general scheduling algorithms discussed in this thesis to the structure of the problem. We point out several future research directions, as follows:

- *Incorporation of prediction information*

  For many CPS applications, it is possible to obtain certain predictions about future system modes and other dynamics, perhaps within a certain time period in the near future. For example, for power systems, it is typically assumed that certain load predictions or renewable generation can be obtained, using historical data or weather predictions. It is possible to utilize such information to improve performance, such as reduction in delay. It is an interesting and challenging research direction to generalize the scheduling schemes in this thesis with prediction information, and compare its behavior and performance with existing research results, such as computationally expensive dynamic programming [88] or heuristic model predictive control methods [89].

- *Distributed implementation*

  Distributed implementations are crucial for certain CPS applications, in particular for the ones without a central coordination entity. For the general scheduling problem with PhyNet

considered in this thesis, it is very promising to develop distributed algorithms, due to the graph sparsity of the physical plant. The detailed design and analysis, on the other hand, may depend heavily on the specific structure of the application. For example, for the simplex scheduling in wireless networks in Chapter 4, the distributed scheduling is implemented with a combination of average consensus and distributed CSMA mechanism.

- *Delay and QoS issues*

  The analyses in this thesis focuses on asymptotic throughput performance, which are based on a stability approach, assuming that all buffers have infinite capacity. Such an assumption may not be true for certain CPS applications, where the buffer may have only finite capacity. Thus, it is also very important to provide rigorous guarantees on delay performance, or other metrics with finite buffers, for these applications. It is an important future work to extend the design and analysis of the scheduling algorithms to address the delay and QoS issues.

APPENDIX **A**

ANALYSIS OF THE HYPERGRAPH INTERFERENCE MODEL FOR WIRELESS NETWORKS

In Section 2.4, we introduced a hypergraph interference model for packet scheduling in wireless networks, as one example of the physical graph model for the general scheduling problem in CPS. Since the hypergraph model is an approximation of the SINR model, this chapter provided quantitative analysis of the modeling accuracy using random networks. Whereas the main purpose of this chapter is to analyze the approximation accuracy versus model complexity tradeoff for the hypergraph interference model, we hope that the same modeling, analysis and design philosophy can be also extended to other CPS applications with physical factor graph approximations.

A.1 OUTAGE ANALYSIS OF THE HYPERGRAPH MODEL

The hypergraph interference model allows more accurate and flexible modeling and control of interference, as compared to the binary interference graph model. In this section, we demonstrate the modeling accuracy of the locally constructed hypergraph model by analyzing its outage probability in random infinite networks, where the nodes form a homogeneous Poisson Point Process (PPP) [90]. We first describe the random network model.

A.1.1 Random Network Model

We consider the Poisson random network model [91], where the set of *contending* nodes form a homogeneous PPP on an infinite two dimensional plane. This model is widely used in the literature of wireless network analysis, since it is tractable, allowing valuable insights into the behavior of large-scale networks. By the Slivnyak's theorem [90], we assume, without loss of generality, that there is a receiver placed at the origin. We further assume that all transmitting nodes transmit with equal power ρ, as is common in 802.11 networks. We assume that the channel is subject to Rayleigh fading. Thus, the received signal power at the center receiver can be expressed as

$$P_0 = \rho h_0 d_0^{-a}, \tag{A.1}$$

where h_0 is the power fading coefficient, which is exponentially distributed with mean 1, d_0 is the length of the center link, and a is the path loss exponent. We assume that SINR is an appropriate metric of performance, and allow the system to be Direct-Sequence Spread Spectrum (DSSS), due to its capability in handling non-trivial levels of multiuser interference in wireless networks. Thus, a packet is received successfully at the center receiver if

$$\frac{\rho h_0 d_0^{-a}}{N_0 + \sum_{j\in\sigma} \rho h_j \|x_j\|^{-a}} \geq \frac{\theta}{M}, \tag{A.2}$$

where N_0 is the received noise power over the entire bandwidth, σ is the set of transmitting links, x_j is the location of the transmitter of a transmitting link j, and M is the spreading factor of DSSS ($M = 1$ in non-spread spectrum systems).

Due to the interference constraint, the set of actual scheduled transmitters in σ is a subset of the contending node set. In fact, the distribution of the transmitting nodes is quite complicated, which depends on various factors, such as the stochastic packet arrival processes, channel fading, and scheduling algorithms. In this paper, in order to make the analysis tractable, we apply an approximation by assuming that the set of transmitting nodes is also a PPP with a smaller density μ, which is obtained by proper 'thinning' of the original PPP. Note that, strictly speaking, the set

of transmitting nodes should be separated by a certain distance, in which case a hard-core point process [90] is more suitable. However, it has been observed that the PPP model can still achieve very accurate approximation [91] on the distribution of the interference, especially when the guard zone sizes are relatively small. This has also been verified by simulation results, in the case of graph interference models (see details in [50]). We next analyze the outage performance under the approximate PPP model.

A.1.2 Outage Analysis

In order to explore the accuracy of the hypergraph model, we assume that the transmission density μ under the hypergraph model is as follows. A hypergraph with maximum hyperedge size K can always guarantee that the following approximate 'local' outage probability P_{out}^{l} at the center receiver is bounded by

$$P_{\text{out}}^{l} = \mathbb{P}\left(\frac{\rho h_0 d_0^{-a}}{N_0 + \sum_{i=1}^{K-1} \rho h_{[i]} \|x_{[i]}\|^{-a}} < \frac{\theta}{M}\right) \leq \epsilon, \tag{A.3}$$

where ϵ is a positive constant, $x_{[i]}$ is the location of the i-th nearest transmitting node, and $h_{[i]}$ is its corresponding power fading coefficient. This is because the hypergraph model approximates the total interference by the sum interference from the nearest $K-1$ transmitters. Further, note that such an outage bound can be easily achieved by a hypergraph with maximum hyperedge size K, since if (A.3) fails to hold for a particular transmitting set consisting of K links, one can simply form a hyperedge to exclude such a transmission scenario. Finally, note that by choosing an appropriate outage bound ϵ, the transmission density μ can be controlled.

Since the approximated 'local' outage probability P_{out}^{l} only considers a subset of interfering links, we are interested in its approximation accuracy with respect to the true outage probability at the center receiver, which is defined as

$$P_{\text{out}} = \mathbb{P}\left(\frac{\rho h_0 d_0^{-a}}{N_0 + \sum_{i=1}^{\infty} \rho h_{[i]} \|x_{[i]}\|^{-a}} < \frac{\theta}{M}\right). \tag{A.4}$$

The key result of this section is the following theorem, which gives closed-form solutions to the outage probabilities P_{out}^{l} and P_{out}.

Theorem A.1.1. *The outage probabilities P_{out}^{l} and P_{out} can be expressed as follows:*

$$P_{out}^{l} = 1-\exp(-\frac{\theta}{M\eta})\int_{0}^{\infty}\frac{2(\mu\pi x^{2})^{K}}{x\Gamma(K)}\Psi(x)e^{-\mu\pi x^{2}}dx \tag{A.5}$$

$$P_{out} = 1-\exp(-\frac{\theta}{M\eta}-\mu\pi d_{0}^{2}\Big(\frac{\theta}{M}\Big)^{\frac{2}{a}}\frac{2\pi/a}{\sin(2\pi/a)}) \tag{A.6}$$

where $\eta=\rho d_{0}^{-a}/N_{0}$ is the Signal to Noise Ratio (SNR) at the center receiver, and

$$\Psi(x)=\big(\int_{0}^{x}\frac{2Mr^{a+1}}{x^{2}(Mr^{a}+d_{0}^{a}\theta)}dr\big)^{K-1}. \tag{A.7}$$

In order to prove the theorem, we first need to prove two lemmas. Define the 'local' interference contributed by the nearest $K-1$ transmitting nodes $I_{\text{loc}}(K-1)$ as follows:

$$I_{\text{loc}}(K-1)=\sum_{i=1}^{K-1}\rho h_{[i]}l(\|x_{[i]}\|). \tag{A.8}$$

We have the following lemma describing the distribution of $I_{\text{loc}}(K-1)$:

Lemma A.1.1. *The Moment Generating Function (MGF) of $I_{loc}(K-1)$ can be expressed as follows:*

$$\Phi_{I_{loc}(K-1)}(s)=\int_{0}^{\infty}\frac{2(\mu\pi x^{2})^{K}}{x\Gamma(K)}\Psi(x)e^{-\mu\pi x^{2}}dx, \tag{A.9}$$

where $\Gamma(K)$ is the standard Gamma function

$$\Gamma(K)=\int_{0}^{\infty}x^{K-1}e^{-x}dx. \tag{A.10}$$

Proof: Denote $R_{k}=\|x_{[k]}\|$ as the short-hand notation for the distance of the k-th nearest transmitting node. The MGF of $I_{\text{loc}}(K-1)$, conditioning on the event that $R_{K}=r_{K}$, can be expressed as follows:

$$\mathbb{E}(e^{sI_{\text{loc}}(K-1)}|R_{K}=r_{K}) = \mathbb{E}(e^{s\sum_{i=1}^{K-1}\rho h_{[i]}l(R_{i})}|R_{K}=r_{K}) \tag{A.11}$$

$$\overset{(a)}{=} \big(\int_{0}^{r_{K}}\frac{2r}{r_{K}^{2}}\mathbb{E}(e^{s\rho hr^{-a}})\mathrm{d}r\big)^{K-1} \tag{A.12}$$

$$\overset{(b)}{=} \big(\int_{0}^{r_{K}}\frac{2r^{a+1}}{r_{K}^{2}(r^{a}-\rho s)}\mathrm{d}r\big)^{K-1} \tag{A.13}$$

where step (a) is because, conditioned there being $K-1$ nodes in the disk centered at the origin with radius r_K, these $K-1$ nodes are independently and uniformly distributed inside the disk, due to the property of PPP [90]. Step (b) is because the fading coefficient h is exponentially distributed with mean 1. Further, according to [92], the distribution of R_K is as follows:

$$f_{R_K}(x) = \frac{2(\mu\pi x^2)^K}{x\Gamma(K)} e^{-\mu\pi x^2}, \tag{A.14}$$

and therefore, the lemma holds after taking the expectation with respect to $f_{R_K}(x)$. ■

Similarly, denote the total interference received at the center receiver as $I_{\text{tot}} = \sum_{i=1}^{\infty} l(\|x_{[i]}\|)$. The following lemma describes the distribution of I_{tot}.

Lemma A.1.2. *The MGF of the total interference I_{tot} is*

$$\Phi_{I_{tot}}(s) = \exp\big(-\mu\pi(-s\rho)^{\frac{2}{a}} \frac{2\pi/a}{\sin(2\pi/a)}\big). \tag{A.15}$$

Proof: This is a standard result. See, for example, [91]. ■

Based on the above lemmas, we are now able to prove the theorem.

Proof of Theorem A.1.1: By definition, we calculate the local outage probability as

$$\begin{aligned}
P_{\text{out}}^{l} &= \mathbb{P}\Big(\frac{\rho h_0 d_0^{-a}}{N_0 + I_{\text{loc}}(K-1)} < \frac{\theta}{M}\Big) && \text{(A.16)}\\
&= \mathbb{P}\Big(h_0 < \frac{d_0^a\theta}{M\rho}\big(N_0 + I_{\text{loc}}(K-1)\big)\Big) && \text{(A.17)}\\
&\overset{(a)}{=} \mathbb{E}_{I_{\text{loc}}(K-1)}\Big\{\mathbb{P}\Big(h_0 < \frac{d_0^a\theta}{M\rho} \times \big(N_0 + I_{\text{loc}}(K-1)\big)|I_{\text{loc}}(K-1)\Big)\Big\} && \text{(A.18)}\\
&\overset{(b)}{=} 1 - \mathbb{E}_{I_{\text{loc}}(K-1)}\{\exp\big(-\frac{d_0^a\theta}{M\rho} I_{\text{loc}}(K-1)\big)\} \times \exp(-\frac{d^a\theta}{M\rho}N_0) && \text{(A.19)}\\
&\overset{(c)}{=} 1 - \Phi_{I_{\text{loc}}(K-1)}(-\frac{d_0^a\theta}{M\rho})\exp(-\frac{d_0^a\theta}{M\rho}N_0), && \text{(A.20)}
\end{aligned}$$

where step (a) follows from the law of total probability, step (b) is because the random variable h_0 is exponentially distributed with mean 1, and step (c) follows from the definition of the MGF. Thus, the claim holds from noting that $\eta = \rho d_0^{-a}/N_0$ and applying the result in (A.9).

Now, using a similar argument, the true outage probability P_{out} can be expressed as follows:

$$\begin{aligned} P_{\text{out}} &= \mathbb{P}\left(\frac{\rho h_0 d_0^{-a}}{N_0 + I_{\text{tot}}} < \frac{\theta}{M}\right) && \text{(A.21)} \\ &= \mathbb{P}\big(h_0 < \frac{d_0^a \theta}{M\rho}(N_0 + I_{\text{tot}})\big) && \text{(A.22)} \\ &= 1 - \mathbb{E}_{I_{\text{tot}}}\{\exp\big(-\frac{d_0^a \theta}{M\rho} I_{\text{tot}}\big)\} \exp(-\frac{d_0^a \theta}{M\rho} N_0) && \text{(A.23)} \\ &= 1 - \Phi_{I_{\text{tot}}}(-\frac{d_0^a \theta}{M\rho}) \exp(-\frac{d_0^a \theta}{M\rho} N_0), && \text{(A.24)} \end{aligned}$$

from which the claim holds after applying (A.15). ■

A.2 NUMERICAL RESULTS

We now illustrate the interference approximation accuracy of the hypergraph model by comparing the above two outage probabilities using numerical calculations.

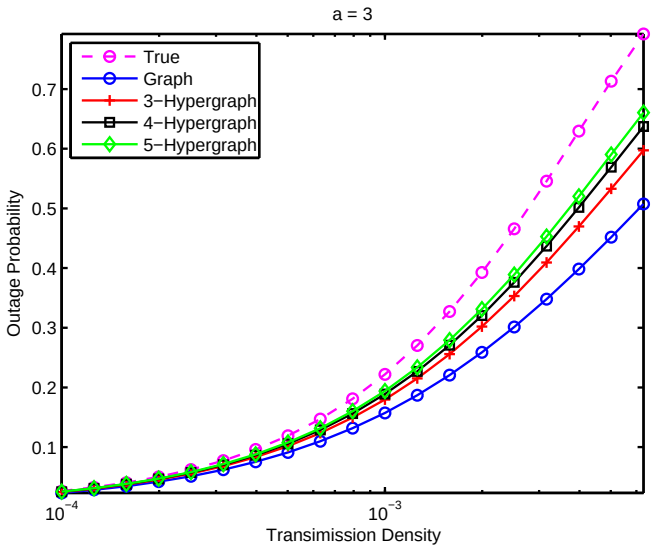
A.2.1 Infinite Random Networks

We next calculate the two outage probabilities in Theorem A.1.1. By choosing parameters as in Table A.1, we plot the numerical results in Fig. A.1, where the outage probabilities are shown in both cases, as functions of the transmission density μ, according to (A.5) and (A.6), under different path loss exponents. In the figure, 'K-Hypergraph' refers to the hypergraph whose maximum hyperedge size is K. Note that since the hypergraph model always underestimates the interference, we have $P_{\text{out}}^l \leq P_{\text{out}}$ for all transmission densities.

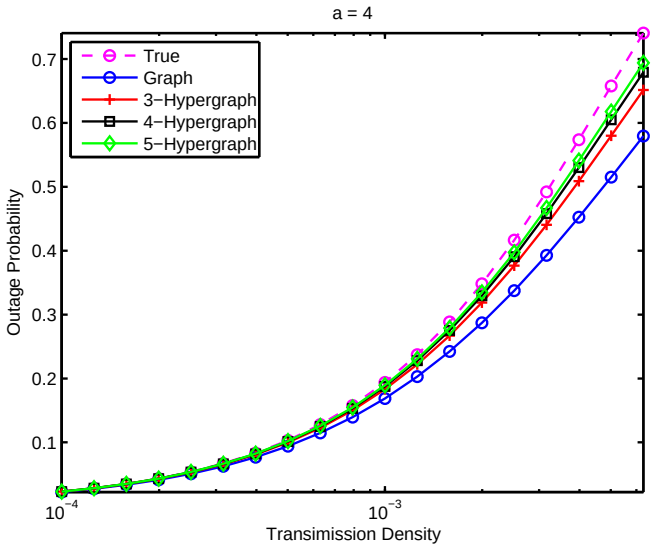
We have the following remarks:

- *Approximation Accuracy*

  Compared to the graph model, the hypergraph model always approximates the true outage probability with better accuracy. For example, when $\mu = 10^{-3}$ and $a = 3$, the true outage probability is around 0.22. However, the outage probability approximation by the graph model is only around 0.15. Therefore, roughly speaking, around 30% of the outage events



(a)



(b)

Figure A.1: The numerical results of outage calculations for the infinite two dimensional random wireless networks with Rayleigh fading. (a) shows the case with the path loss exponent $a = 3$, and (b) shows the case with $a = 4$.

are ignored by the graph, due to its binary interference nature. In this case, the 4-Hypergraph has a better approximation of around 0.19. Thus, by considering the sum interference, the hypergraph model can effectively capture more outage events, and therefore reduces the outage

Table A.1: Parameters for Numerical Calculations

Symbol	Description	Value
d_0	Center link length	10m
θ	Target SINR	$3 = 4.77$dB
η	SNR	20dB
M	Spreading factor	16

probability.

- *Accuracy versus Complexity*

  The approximation accuracy of the hypergraph has a 'diminishing returns' property, which can be seen by observing the fact that, the outage approximation error improvement decreases as the maximum hyperedge size K increases. On the other hand, the construction complexity of the hypergraph increases exponentially in K. Thus, one can trade-off some approximation accuracy by only considering properly small hyperedge sizes, so that the sum interference is approximated with low complexity.

- *Effect of Path Loss Exponent*

  The modeling accuracy of both graph and hypergraph models improves when the pass loss exponent gets larger. In particular, when $a = 4$ and $P_{\text{out}} = 0.3$, all hypergraphs can capture above 95% outage events, and the graph can model around 80% outage events, as can be seen by computing the ratio $P^l_{\text{out}}/P_{\text{out}}$. On the other hand, for the case $a = 3$ and $P_{\text{out}} = 0.3$, the ratio is only about 85% for hypergraphs and about 70% for the graph. Such improvement with larger values of a is because, when a grows larger, the contributions of the far-away interferers are much smaller as compared to the near-by interferers, and therefore, the local interference approximation becomes more accurate.

A.2.2 A Finite Random Network

We next consider the performance of hypergraph model in a finite random network, where the topology is shown in Fig. A.2. In the figure, 40 links are uniformly distributed over a two dimensional plane. The square nodes are transmitters, and the round nodes are receivers. We assume that the link lengths are equal. We further assume that the effect of fading is properly handled using diversity techniques, such that all random coefficients $\{h_i\}$ in (A.2) take the constant value 1. The SNR at the receivers are the same for all links. The parameters used in the simulation are shown in Table A.1.

The hypergraph is generated according to the description in Section 2.4, with the modification that (2.16) is replaced with

$$\frac{S_i}{N_i + \sum_{s=1}^{k-1} I_{i_s i}} < \frac{\theta}{M}, \tag{A.25}$$

due to the DSSS physical layer. Further, we assume that no superset of a hyperedge is a hyperedge, so that the hypergraph specification is not redundant.

In the simulation, we assume that the packet arrival processes are i.i.d, with Bernoulli distribution and a uniform arrival rate. We simulate both hypergraph (which includes graph) and the global SINR based scheduling algorithms. We consider random maximal scheduling for the hypergraph case, which adds the links to the schedule according to a randomly generated order in each time slot, such that a back-logged link is scheduled if there is no hyperedge constraint violation when it is being considered. We also consider the random maximal scheduling under the global SINR, which we denote as SINR-MS. As an upper bound, we simulate the performance of SINR based LQF scheduling (SINR-LQF) in [93], which adds the links according to the queue lengths order, subject to the physical SINR constraint (A.2). Finally, the packet reception at a transmitting link i is assumed to fail if the true SINR constraint (A.2) is violated.

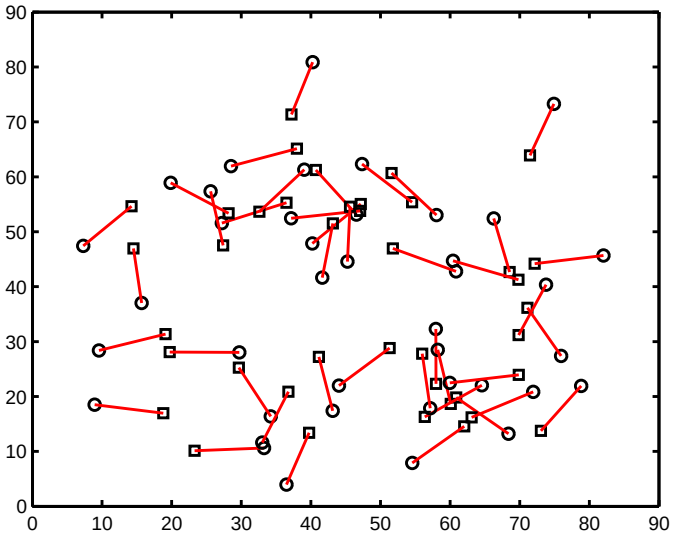


Figure A.2: The topology of the random network with 40 links used for simulation. The square nodes are transmitters, and the round nodes are receivers.

A.2.2.1 Throughput

The total queue lengths under different arrival rates are shown in Fig. A.3. The results are averaged over 30 independent simulations, where each simulation consists of 10^4 time slots. One can detect the boundary of the stability region (the maximum uniform throughput) by identifying the point at which the total queue length begins to increase sharply. For example, in the case of $a = 3$ and $\beta = 4$dB, the graph model achieves a maximum uniform rate of 0.22, the hypergraph models achieve about 0.24, the SINR-MS achieves around 0.26, and the SINR-LQF has the largest rate, which is around 0.30. In the case of random maximal scheduling, the SINR based scheduling algorithms achieve around 10% throughput gain over hypergraph based algorithms when $a = 3$ and $\beta = 4$dB. The gain is around 5% when $a = 4$ and $\beta = 4$dB. Such throughput gain is mainly because of the *perfect accuracy* of the SINR model, which results in zero packet collision. On the other hand, this is achieved at the expense of *network-wide user node coordination*, due to the global nature of the SINR model. In this sense, the hypergraph based schedulers are more attractive, due

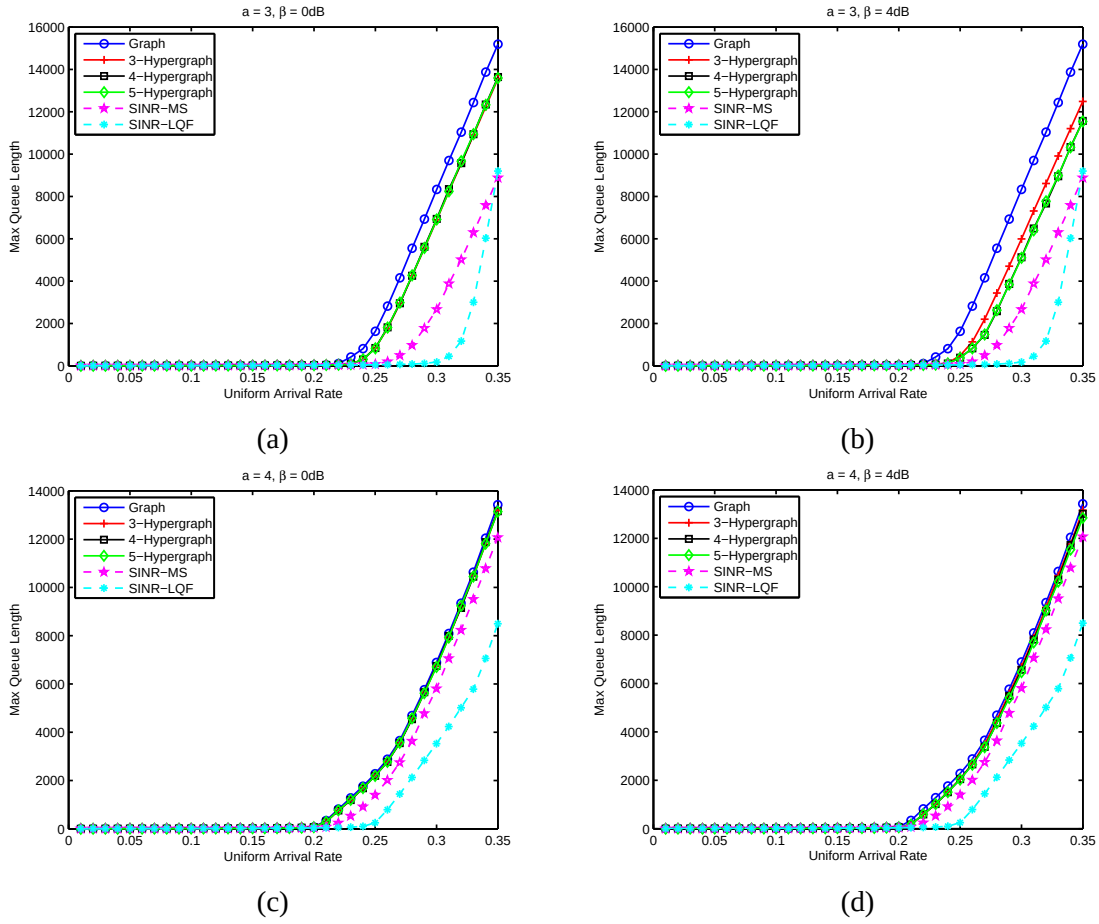


Figure A.3: The simulation results of the maximum total queue lengths in the 40-link random wireless network, with the path loss exponent a and the threshold β values shown in each figure.

to the localized user coordination during scheduling. The better throughput performance of SINR-LQF over SINR-MS is also expected, since the queue lengths order information is used in the former case, which requires higher scheduling complexity.

In all cases, the hypergraph based scheduling algorithms achieve larger throughput than the graph based scheduling. Note that due to the construction procedure in (A.25), the set of hyperedges is always a superset of the set of edges, and therefore, it may seem like the hypergraph 'should' achieve a smaller capacity than the graph, due to the more restricted rules. However, since the graph model is a binary approximation, the actual throughput is reduced by the packet collisions

caused by its 'aggressive' transmissions. Thus, even though the hypergraph models are relatively 'conservative', since they place more constraints according to the sum interference, overall they can still achieve a better throughput as compared to the graph model.

Finally, by comparing the throughput results under different β, one can easily observe the 'accuracy versus complexity' trade-off for hypergraph models. In the simulation results, the throughput performance for hypergraph schedulers improves with larger β, due to more accurate interference approximation by adding more links in $\mathcal{L}_i$, so that the number of packet collisions is reduced. Note that the throughput gain also depends on a, where larger a implies more accurate interference approximation for the same β, since the contribution from far-away links gets smaller.

A.2.2.2 Outage Probability

The simulation results of average outage probabilities are shown in Fig. A.4 with different arrival rates, under different path loss exponent a and threshold β values. Note that the results of both SINR based scheduling are not shown in the figure, as they both have zero outage.

Remarks:

- *Outage Probability*

  The hypergraph model achieves a significant outage probability reduction, as compared to the graph model. For example, when $a = 3$ and $\beta = 4$dB, the average outage probability for the 3-Hypergraph is around 0.02 under arrival rate 0.2. On the other hand, under the graph model it is around 0.05. Thus, by modeling the *sum* interference, the hypergraph model can reduce the packet collisions very effectively. Note that the outage probability curves have slower slopes when the arrival rates are large. This corresponds to the case when the network is unstable. In such a case, the number of packet transmissions is less sensitive to the increase in arrival rates.

- *Outage Capacity*

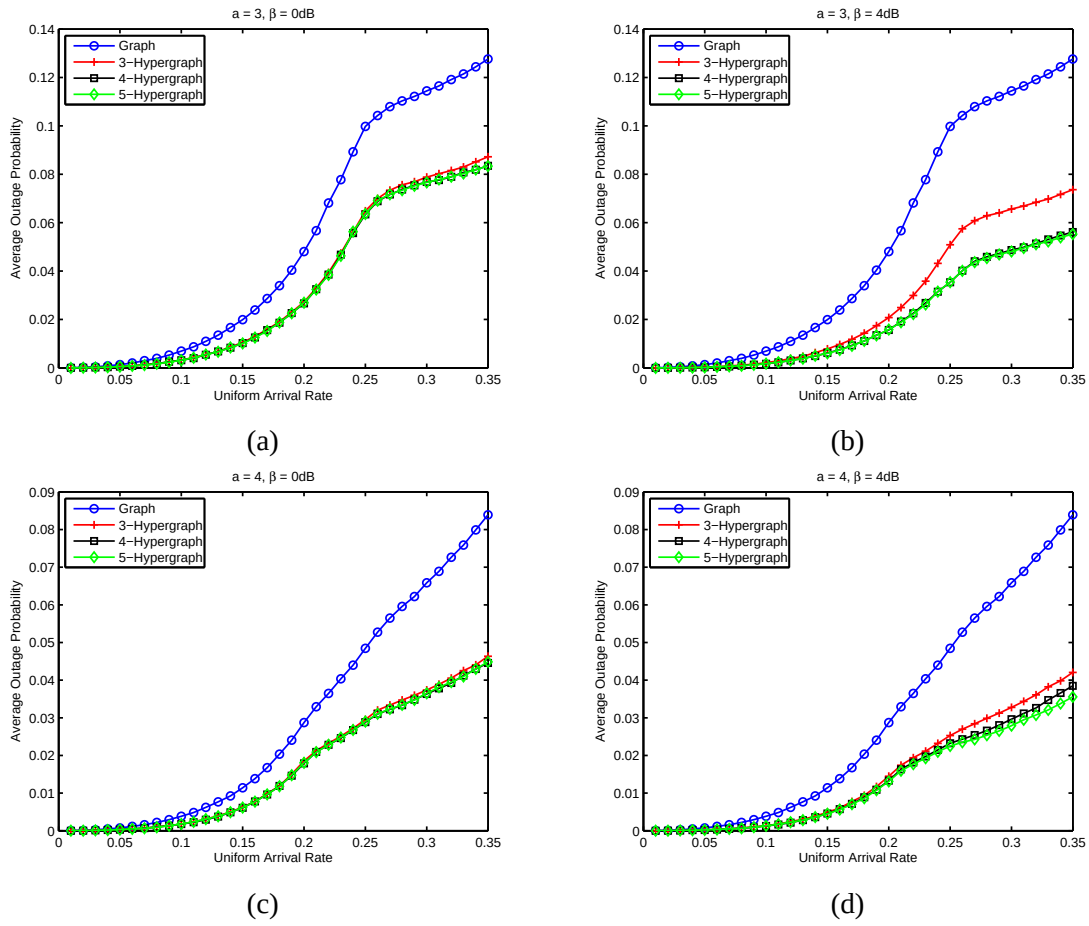


Figure A.4: The simulation results of average outage probability in the 40-link random wireless network, with the path loss exponent a and the threshold β values shown in each figure.

If we consider the *outage capacity*, i.e., the maximum achievable rate under certain outage probability constraint, the hypergraph interference models can have much larger gain as compared to the graph model. For example, under the outage probability constraint of 0.02 when $a = 3$ and $\beta = 4$dB, the 4-Hypergraph can support a maximum rate of around 0.2, whereas the graph model can only achieve about 0.15. Thus, by considering the sum interference, the hypergraph model achieves around 30% increase in the outage capacity as compared to the graph model.

- *Accuracy versus Complexity*

The outage probabilities of hypergraph models decrease when the threshold β increases, due to the improved approximation accuracy by considering more links in a link's neighborhood. For example, when $a = 3$ and $\beta = 0$dB, all hypergraph models have an outage probability of 0.03 under arrival rate 0.2, as compared to around 0.02 when $\beta = 4$dB. On the other hand, the threshold β has no effect on the graph model, due to the binary interference nature. Further, note that when β is small, all hypergraph models achieve very similar outage probability results, in which case the 3-Hypergraph is more attractive, due to its lower coordination complexity. In general, increasing the threshold β can effectively reduce the outage probability by considering the interference from farther transmitting links, but at the expense of more coordination overheads among links. Finally, the reduction in the outage probability decreases as the hypergraph sizes increases. This 'diminishing marginal returns' property, again, confirms our observation that in wireless networks, the majority of the interference is from a few nearby transmitting links. Therefore, a hypergraph with a small hyperedge size (e.g., 4-Hypergraph) can model the interference with good accuracy.

- *Effect of Path Loss Exponent*

  When the path loss exponent a gets larger, both graph and hypergraph models have smaller outage probabilities. Further, the gap between these two models is also smaller. This is the same conclusion as the numerical results for infinite random networks. The intuition is that, the interference signals get more attenuation as a gets larger, and therefore, is more likely to be dominated by a few nearby transmitting links, since the faraway transmissions are attenuated more severely than the nearby transmissions. Thus, the accuracy of both the graph and hypergraph models become better, and the difference between the two is also smaller.

APPENDIX **B**

PROOFS IN CHAPTER 3

B.1 CONSTRUCTION OF FLUID LIMITS

This section presents a brief introduction to the theory of fluid limits. For details, we urge interested readers to read [74, 75] and the references therein. The construction of a fluid limit is as follows. Given the discrete-time queueing system in Section 3.1, we first obtain a continuous time system by extending the support from $\mathbb{N}$ to $\mathbb{R}_+$ using linear interpolation. For a fixed sample path ω, define the following fluid scaling:

$$g^r(t,\omega) = \frac{g(rt,\omega)}{r}, \tag{B.1}$$

where $r > 0$ is a positive scalar, and the function $g(\cdot)$ can be $U_i(\cdot), \Lambda_i(\cdot)$, $Y_i(\cdot)$, $Z_i(\cdot)$, $T_s^{\alpha}(\cdot)$, $T_s(\cdot)$ and $\hat{F}_j(\cdot)$. From the Assumption 2.2.1 and Assumption 2.2.2, and the fact that each $\mathcal{A}_i$ is a finite set, it is not difficult to verify that these functions are uniformly Lipschitz-continuous [94]. That is, there is a positive constant $K > 0$ such that

$$|g^r(t+\delta) - g^r(t)| \quad \leq \quad K\delta \tag{B.2}$$

for any r, $t > 0$ and $\delta > 0$. Thus, these functions are equi-continuous. According to the Arzéla-Ascoli Theorem [94], any sequence of functions $\{g^{r_n}(t)\}_{n=1}^{\infty}$ contains a subsequence $\{g^{r_{n_k}}(t)\}_{k=1}^{\infty}$,

such that w.p.1, we have

$$\lim_{k\to\infty} \sup_{\tau\in[0,t]} |g^{r_{n_k}}(\tau) - \bar{g}(\tau)| = 0 \tag{B.3}$$

where $\bar{g}(t)$ is a uniformly continuous function, and therefore differentiable almost everywhere [94]. We can then define any such limit as a fluid limit.

B.2 PROOF OF LEMMA 3.1.1

Suppose the rate stability does not hold for user i. Then, there is $i \in \mathcal{V}$ and a sequence $\{r_n\}$ such that

$$\lim_{n\to\infty} \frac{U_i(r_n)}{r_n} \geq \epsilon', \tag{B.4}$$

for some $\epsilon' > 0$. Now, as all functions are equi-continuous, according to the construction of fluid limit, we can find a subsequence $\{r_{n_k}\}$, which converges to a fluid limit. Thus, according to (B.4), we have

$$\bar{U}_i(1) \geq \epsilon', \tag{B.5}$$

which contradicts the assumption that $\bar{U}_i(t) = 0$ for all $t > 0$. Thus, we claim that rate stability holds for queue $U_i(n)$ in the original stochastic system.

B.3 PROOF OF LEMMA 3.1.2

Since the lemma claims the result holds for both algorithms, we prove them separately. We first prove the case with Algorithm 3.1.1.

Proof: (Part I)

Let $t > 0$, $s \in \mathcal{S}$ and $\alpha \in \mathcal{C}(s)$ be given. Assume that there is $\alpha' \in \mathcal{C}(s)$ such that

$$\sum_{i\in\mathcal{V}} \bar{U}_i(t)\alpha_i - \beta \sum_{j\in\mathcal{J}} \bar{\Phi}_j(t) f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}) < \sum_{i\in\mathcal{V}} \bar{U}_i(t)\alpha'_i - \beta \sum_{j\in\mathcal{J}} \bar{\Phi}_j(t) f_j(\alpha'_{\mathcal{N}_j}; s_{\mathcal{N}_j}). \tag{B.6}$$

Since all functions in the fluid limit are uniformly continuous, there is $\epsilon' > 0$ and $\delta > 0$ such that

for any $\tau \in (t-\delta, t+\delta)$, we have

$$\sum_{i\in\mathcal{V}} \bar{U}_i(\tau)\alpha_i - \beta \sum_{j\in\mathcal{J}} \bar{\Phi}_j(\tau) f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}) \leq \sum_{i\in\mathcal{V}} \bar{U}_i(\tau)\alpha'_i - \beta \sum_{j\in\mathcal{J}} \bar{\Phi}_j(\tau) f_j(\alpha'_{\mathcal{N}_j}; s_{\mathcal{N}_j}) - \epsilon'. \quad \text{(B.7)}$$

Thus, consider any convergent subsequence for the fluid limit. There is K such that for any $k \geq K$, we have

$$\sum_{i\in\mathcal{V}} U_i^{r_{n_k}}(\tau)\alpha_i - \beta \sum_{j\in\mathcal{J}} \Phi_j^{r_{n_k}}(\tau) f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}) \leq \sum_{i\in\mathcal{V}} U_i^{r_{n_k}}(\tau)\alpha'_i - \beta \sum_{j\in\mathcal{J}} \Phi_j^{r_{n_k}}(\tau) f_j(\alpha'_{\mathcal{N}_j}; s_{\mathcal{N}_j}) - \frac{\epsilon'}{2} \quad \text{(B.8)}$$

for any $\tau \in (t-\delta, t+\delta)$. According to the definition of fluid scaling, this implies that

$$\sum_{i\in\mathcal{V}} U_i(\tau)\alpha_i - \beta \sum_{j\in\mathcal{J}} \Phi_j(\tau) f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}) \leq \sum_{i\in\mathcal{V}} U_i(\tau)\alpha'_i - \beta \sum_{j\in\mathcal{J}} \Phi_j(\tau) f_j(\alpha'_{\mathcal{N}_j}; s_{\mathcal{N}_j}) - \frac{r_{n_k}\epsilon'}{2} \quad \text{(B.9)}$$

for any $\tau \in (r_{n_k}(t-\delta), r_{n_k}(t+\delta))$. Thus, according to the augmented max-weight scheduler in (3.5), the control action α is never chosen during the time period $(r_{n_k}(t-\delta), r_{n_k}(t+\delta))$, from which we conclude that

$$\left(T_s^\alpha\right)^{r_{n_k}}(t+\delta) = \left(T_s^\alpha\right)^{r_{n_k}}(t-\delta), \forall k \geq K, \quad \text{(B.10)}$$

which further implies that, after taking $k \to \infty$, we have

$$\bar{T}_s^\alpha(t+\delta) = \bar{T}_s^\alpha(t-\delta). \quad \text{(B.11)}$$

Thus, the lemma follows from the fact that the $\dot{\bar{T}}_s^\alpha(t) \geq 0$. ■

We now consider the case with Algorithm 3.1.2.

Proof: (Part II)

Let $t > 0$, $s \in \mathcal{S}$ and $\alpha \in \mathcal{C}(s)$ be given. According to the assumption, there is a schedule $\tilde{\alpha}$

and $\epsilon' > 0$ such that

$$\sum_{i\in\mathcal{V}} \bar{U}_i(t)\alpha_i - \beta \sum_{j\in\mathcal{J}} \bar{\Phi}_j(t) f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}) \leq \sum_{i\in\mathcal{V}} \bar{U}_i(t)\tilde{\alpha}_i - \beta \sum_{j\in\mathcal{J}} \bar{\Phi}_j(t) f_j(\tilde{\alpha}_{\mathcal{N}_j}; s_{\mathcal{N}_j}) - \epsilon'. \tag{B.12}$$

Now we define $\mathcal{B}$ as the set of schedules such that $\alpha' \in \mathcal{B}$ implies that $\alpha' \in \mathcal{C}(s)$, and that

$$\sum_{i\in\mathcal{V}} \bar{U}_i(t)\alpha_i - \beta \sum_{j\in\mathcal{J}} \bar{\Phi}_j(t) f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}) \leq \sum_{i\in\mathcal{V}} \bar{U}_i(t)\alpha'_i - \beta \sum_{j\in\mathcal{J}} \bar{\Phi}_j(t) f_j(\alpha'_{\mathcal{N}_j}; s_{\mathcal{N}_j}) - \epsilon'. \tag{B.13}$$

We can choose ϵ' sufficiently small so that the cardinality of $\mathcal{B}$ is maximized. Thus, $\mathcal{B}$ is the set of schedules whose weights are larger than α by at least ϵ'. Note that $\mathcal{B}$ is not empty. Further, since all functions in the fluid limit are uniformly continuous, there is $\delta > 0$ such that for any $\tau \in (t-\delta, t+\delta)$ and $\alpha' \in \mathcal{B}$, we have

$$\sum_{i\in\mathcal{V}} \bar{U}_i(\tau)\alpha_i - \beta \sum_{j\in\mathcal{J}} \bar{\Phi}_j(\tau) f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}) \leq \sum_{i\in\mathcal{V}} \bar{U}_i(\tau)\alpha'_i - \beta \sum_{j\in\mathcal{J}} \bar{\Phi}_j(\tau) f_j(\alpha'_{\mathcal{N}_j}; s_{\mathcal{N}_j}) - \frac{\epsilon'}{2}. \tag{B.14}$$

Thus, consider any convergent subsequence for the fluid limit. There is K such that for any $k \geq K$, we have

$$\sum_{i\in\mathcal{V}} U_i^{r_{n_k}}(\tau)\alpha_i - \beta \sum_{j\in\mathcal{J}} \Phi_j^{r_{n_k}}(\tau) f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}) \leq \sum_{i\in\mathcal{V}} U_i^{r_{n_k}}(\tau)\alpha'_i - \beta \sum_{j\in\mathcal{J}} \Phi_j^{r_{n_k}}(\tau) f_j(\alpha'_{\mathcal{N}_j}; s_{\mathcal{N}_j}) - \frac{\epsilon'}{4} \tag{B.15}$$

for any $\tau \in (t-\delta, t+\delta)$ and $\alpha' \in \mathcal{B}$. According to the definition of fluid scaling, this implies that

$$\sum_{i\in\mathcal{V}} U_i(\tau)\alpha_i - \beta \sum_{j\in\mathcal{J}} \Phi_j(\tau) f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}) \leq \sum_{i\in\mathcal{V}} U_i(\tau)\alpha'_i - \beta \sum_{j\in\mathcal{J}} \Phi_j(\tau) f_j(\alpha'_{\mathcal{N}_j}; s_{\mathcal{N}_j}) - \frac{r_{n_k}\epsilon'}{4} \tag{B.16}$$

for any $\tau \in (r_{n_k}(t-\delta), r_{n_k}(t+\delta))$. This implies that, for sufficiently large k, when Algorithm 3.1.2 chooses any schedule in $\mathcal{B}$ during the time interval $(r_{n_k}(t-\delta), r_{n_k}(t+\delta))$, the schedules will never leave the set $\mathcal{B}$ if the system mode is s, since $\mathcal{B}$ is maximal. Therefore, the schedule α will never be chosen again during the same time interval. Now define

$$\Delta_k = T_s^{\alpha}(r_{n_k}(t+\delta)) - T_s^{\alpha}(r_{n_k}(t-\delta)) \tag{B.17}$$

as the total number of time slots that schedule α is chosen during the time interval $(r_{n_k}(t-\delta), r_{n_k}(t+\delta))$ when the system mode is s. We will show that

$$\mathbb{P}(\limsup_{k\to\infty} \left(\frac{\Delta_k}{2r_{n_k}\delta}\right) \geq \delta_0) = 0 \tag{B.18}$$

for any $\delta_0 > 0$, from which we can conclude that the following is true:

$$\begin{aligned} \mathbb{P}(\limsup_{k\to\infty} \left(\frac{\Delta_k}{2r_{n_k}\delta}\right) = 0) &= 1 - \mathbb{P}(\cup_{m=1}^{\infty}\{\limsup_{k\to\infty} \left(\frac{\Delta_k}{2r_{n_k}\delta}\right) \geq \frac{1}{m}\}) &\text{(B.19)}\\ &= 1, &\text{(B.20)} \end{aligned}$$

which implies that, in the fluid limit, we have

$$\bar{T}_s^{\alpha}(t+\delta) = \bar{T}_s^{\alpha}(t-\delta), \tag{B.21}$$

from which the lemma holds. Now we prove (B.18). We now fix the system mode s and a schedule $\alpha' \in \mathcal{B}$. Define the 'hitting time' H_k as the total number of time slots that have passed since time slot $r_{n_k}(t-\delta)$ before Algorithm 3.1.2 randomly generates schedule α' for the first time when the system mode is at s. Note that H_k only counts the time slots when the system mode is s. Thus, once α' is generated for the first time, schedule α is never chosen during for the rest of the time interval, as the schedules will be restricted to the set $\mathcal{B}$. We now have

$$\Delta_k \leq H_k, \text{w.p.1}, \tag{B.22}$$

and it is sufficient to prove that

$$\mathbb{P}(\limsup_{k\to\infty}\Big(\frac{H_k}{2r_{n_k}\delta}\Big)\geq\delta_0)=0. \tag{B.23}$$

Without loss of generality, we assume that $r_{n_k}\geq k/2\delta\delta_0$. Define the event A_k as

$$A_k=\{\omega:\frac{H_k}{2r_{n_k}\delta}\geq\delta_0\}. \tag{B.24}$$

We have

$$\begin{aligned}\mathbb{P}(A_k) &\overset{(a)}{\leq} (1-\epsilon_0)^{2r_{n_k}\delta\delta_0} &\text{(B.25)}\\ &\leq (1-\epsilon_0)^k, &\text{(B.26)}\end{aligned}$$

where (a) is because of the random generation in Algorithm 3.1.2. Thus, we have

$$\begin{aligned}\sum_{k=1}^{\infty}\mathbb{P}(A_k) &\leq \sum_{k=1}^{\infty}(1-\epsilon_0)^k &\text{(B.27)}\\ &< \infty, &\text{(B.28)}\end{aligned}$$

from which we conclude that $\sum_{k=1}^{\infty}\mathbb{P}(A_k)$ converges. According to the first Borel-Cantelli Lemma [95], we have

$$\mathbb{P}(\limsup_k A_k)=0, \tag{B.29}$$

from which we conclude that (B.23) holds. ■

B.4 PROOF OF LEMMA 3.1.3

Before proving the stability results in the fluid limits, we need to prove some technical lemmas. Firstly, the following lemma shows that all external stochastic processes are deterministic:

Lemma B.4.1. *The following are true for any fluid limit:*

$$\begin{aligned}\dot{\bar{\Lambda}}_i(t)&=\lambda_i \quad \forall i\in\mathcal{V},t>0 &\text{(B.30)}\\ \dot{\bar{F}}_j(t)&=\hat{f}_j^{\star} \quad \forall j\in\mathcal{J},t>0 &\text{(B.31)}\\ \dot{\bar{T}}_s(t)&=\pi_s \quad \forall s\in\mathcal{S},t>0. &\text{(B.32)}\end{aligned}$$

Proof: It is easy to verify (B.30) from the assumption of SLLN in (2.5). (B.31) is because of the assumption in (3.1) and the definition in (3.2). Finally, (B.32) is because of the SLLN assumption in (2.8). ■

The following lemma shows the properties of the idling processes in the fluid limit. That is, the cumulative idling processes remains constant when the queues are nonzero.

Lemma B.4.2. *The following are true for any fluid limit:*

$$\dot{\bar{Y}}_i(t) = 0 \quad \textit{if } \bar{U}_i(t) > 0, \forall i \in \mathcal{V} \tag{B.33}$$

$$\dot{\bar{Z}}_j(t) = 0 \quad \textit{if } \bar{\Phi}_j(t) > 0. \forall j \in \mathcal{J} \tag{B.34}$$

Proof: We only prove the first case. The proof of the second one follows an identical procedure as the first one. Assume that $\bar{U}_i(t) > 0$ for some $i \in \mathcal{V}$ and $t > 0$. Since all functions in the fluid limit are uniformly continuous, we can find $\epsilon > 0$ and $\delta > 0$, such that the following is true:

$$\bar{U}_i(\tau) \geq \epsilon, \forall \tau \in (t - \delta, t + \delta). \tag{B.35}$$

Now, we consider any subsequence which converges to the fluid limit. Due to the definition of uniform convergence on compact sets in (B.3), there is a large constant K such that

$$U_i^{r_{n_k}}(\tau) \geq \frac{\epsilon}{2}, \forall \tau \in (t - \delta, t + \delta), k \geq K. \tag{B.36}$$

Recalling the definition of fluid scaling, this implies that

$$\frac{U_i(r_{n_k}\tau)}{r_{n_k}} \geq \frac{\epsilon}{2}, \forall \tau \in (t - \delta, t + \delta), k \geq K. \tag{B.37}$$

Thus, for large enough k, we have

$$U_i(\tau) \geq \frac{r_{n_k}\epsilon}{2} \geq \alpha_i^{\max}, \forall \tau \in (r_{n_k}(t - \delta), r_{n_k}(t + \delta)), \tag{B.38}$$

where $\alpha_i^{\max}$ is the largest job departure rate in each time slot for user i. Thus, the queue of user i is always non-idling during the time interval $(r_{n_k}(t - \delta), r_{n_k}(t + \delta))$. Therefore, we have

$$Y_i(r_{n_k}(t + \delta)) = Y_i(r_{n_k}(t - \delta)), \tag{B.39}$$

which implies that, after fluid scaling and taking $k \to \infty$, we have

$$\bar{Y}_i(t+\delta) = \bar{Y}_i(t-\delta). \tag{B.40}$$

Finally, the claim holds following the fact that $\bar{Y}_i(t)$ is a non-decreasing function. ■

We are now ready to prove the lemma.

Proof of Lemma 3.1.3: Due to the feasibility assumption of OPT-F, it is well-known that the 'arrival rates' should be inside the convex hull of the departure schedules, i.e.,

$$\lambda_i \leq \sum_{s\in\mathcal{S}} \sum_{\alpha\in\mathcal{C}(s)} \mu_s^\alpha \alpha_i, \forall i \tag{B.41}$$

$$\hat{f}_j^\star \geq \sum_{s\in\mathcal{S}} \sum_{\alpha\in\mathcal{C}(s)} \mu_s^\alpha f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}), \forall j, \tag{B.42}$$

where the set of coefficients $\{\mu_s^\alpha\}$ satisfy

$$\mu_s^\alpha \geq 0, \forall s, \alpha \tag{B.43}$$

$$\sum_{\alpha\in\mathcal{C}(s)} \mu_s^\alpha = 1, \forall s. \tag{B.44}$$

The proof is standard, see for example, [11]. Now, let a fluid limit be given. Define the following Lyapunov function:

$$L(t) = \frac{1}{2}\sum_{i\in\mathcal{V}} (\bar{U}_i(t))^2 + \frac{\beta}{2}\sum_{j\in\mathcal{J}} (\bar{\Phi}_j(t))^2. \tag{B.45}$$

We calculate its drift as follows:

$$
\begin{aligned}
& \dot{L}(t) \\
= \; & \sum_{i\in\mathcal{V}} \bar{U}_i(t)\dot{\bar{U}}_i(t) + \beta \sum_{j\in\mathcal{J}} \bar{\Phi}_j(t)\dot{\bar{\Phi}}_k(t) && \text{(B.46)} \\
= \; & \sum_{i\in\mathcal{V}} \bar{U}_i(t)\Big(-\sum_{s\in\mathcal{S}}\sum_{\alpha\in\mathcal{C}(s)} \dot{\bar{T}}_s^{\alpha}(t)\alpha_i + \lambda_i + \dot{\bar{Y}}_i(t)\Big) \\
& \quad + \beta \sum_{j\in\mathcal{J}} \bar{\Phi}_j(t)\Big(\sum_{s\in\mathcal{S}}\sum_{\alpha\in\mathcal{C}(s)} \dot{\bar{T}}_s^{\alpha}(t) f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}) - \dot{\bar{F}}_j(t) + \dot{\bar{Z}}_i(t)\Big) && \text{(B.47)} \\
\overset{(a)}{=} \; & \sum_{i\in\mathcal{V}} \bar{U}_i(t)\Big(-\sum_{s\in\mathcal{S}}\sum_{\alpha\in\mathcal{C}(s)} \dot{\bar{T}}_s^{\alpha}(t)\alpha_i + \lambda_i\Big) \\
& \quad + \beta \sum_{j\in\mathcal{J}} \bar{\Phi}_j(t)\Big(\sum_{s\in\mathcal{S}}\sum_{\alpha\in\mathcal{C}(s)} \dot{\bar{T}}_s^{\alpha}(t) f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}) - \dot{\bar{F}}_j(t)\Big) && \text{(B.48)} \\
\leq \; & \sum_{i\in\mathcal{V}} \bar{U}_i(t)\Big(-\sum_{s\in\mathcal{S}}\sum_{\alpha\in\mathcal{C}(s)} \dot{\bar{T}}_s^{\alpha}(t)\alpha_i + \sum_{s\in\mathcal{S}}\sum_{\alpha\in\mathcal{C}(s)} \mu_s^{\alpha}\alpha_i\Big) \\
& \quad + \beta \sum_{j\in\mathcal{J}} \bar{\Phi}_j(t)\Big(\sum_{s\in\mathcal{S}}\sum_{\alpha\in\mathcal{C}(s)} \dot{\bar{T}}_s^{\alpha}(t) f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j}) - \sum_{s\in\mathcal{S}}\sum_{\alpha\in\mathcal{C}(s)} \mu_s^{\alpha} f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j})\Big) && \text{(B.49)} \\
= \; & -\sum_{s\in\mathcal{S}}\sum_{\alpha\in\mathcal{C}(s)} \big(\dot{\bar{T}}_s^{\alpha}(t) - \mu_s^{\alpha}\big)\Big(\sum_{i\in\mathcal{V}} \bar{U}_i(t)\alpha_i - \beta \sum_{j\in\mathcal{J}} \bar{\Phi}_j(t) f_j(\alpha_{\mathcal{N}_j}; s_{\mathcal{N}_j})\Big) && \text{(B.50)} \\
\overset{(b)}{\leq} \; & 0. && \text{(B.51)}
\end{aligned}
$$

where (a) is because of Lemma B.4.2, and (b) is because of the max-weight property proved in Lemma 3.1.2. Thus, we have $L(t) = 0$ if $L(0) = 0$, from which the lemma holds. ■

B.5 PROOF OF LEMMA 3.2.1

In order to prove Lemma 3.2.1, we need to prove several technical lemmas first. We first provide a bound on the single slot drift of $L(n)$.

Lemma B.5.1. *The one-slot drift of $L(n)$ satisfies the following under any control action $\alpha(n+1)$:*

$$
\begin{aligned}
\Delta_1 L(n) \; \leq \; & \sum_{i\in\mathcal{V}} (U_i(n) + \zeta_i)(\Lambda_i(n+1) - \alpha_i(n+1)) + \beta \sum_{j\in\mathcal{J}} f_j(\alpha_{\mathcal{N}_j}(n+1); s_{\mathcal{N}_j}(n+1)) \\
& + \sum_{i\in\mathcal{V}} \alpha_i^{\max}\zeta_i + \frac{1}{2}\sum_{i\in\mathcal{V}} (\Lambda_i^{\max} + \alpha_i^{\max})^2 + \sum_{i\in\mathcal{V}} (\alpha_i^{\max})^2. && \text{(B.52)}
\end{aligned}
$$

Proof: For each user $i \in \mathcal{V}$, direct calculation shows that

$$\begin{aligned}
& \frac{1}{2}(U_i(n+1)+\zeta_i)^2 && \text{(B.53)}\\
= \;& \frac{1}{2}\big(U_i(n)-\alpha_i(n+1)\wedge U_i(n)+\Lambda_i(n+1)+\zeta_i\big)^2 && \text{(B.54)}\\
= \;& \frac{1}{2}(U_i(n)+\zeta_i)^2+(U_i(n)+\zeta_i)\big(\Lambda_i(n+1)-\alpha_i(n+1)\wedge U_i(n)\big)\\
& \quad +\frac{1}{2}(\Lambda_i(n+1)-\alpha_i(n+1)\wedge U_i(n))^2 && \text{(B.55)}\\
\overset{(a)}{\leq} \;& \frac{1}{2}(U_i(n)+\zeta_i)^2+(U_i(n)+\zeta_i)\big(\Lambda_i(1)-\alpha_i(n+1)\big)+(\alpha_i^{\max}+\zeta_i)\alpha_i^{\max}\\
& \quad +\frac{1}{2}(\Lambda_i(n+1)-\alpha_i(n+1)\wedge U_i(n))^2 && \text{(B.56)}\\
\leq \;& \frac{1}{2}(U_i(n)+\zeta_i)^2+(U_i(n)+\zeta_i)\big(\Lambda_i(1)-\alpha_i(n+1)\big)+\alpha_i^{\max}\zeta_i\\
& \quad +\frac{1}{2}(\Lambda_i^{\max}+\alpha_i^{\max})^2+(\alpha_i^{\max})^2, && \text{(B.57)}
\end{aligned}$$

where the key step (a) can be verified as follows. When $U_i(n) > \alpha_i(n+1)$, it is obvious that (a) holds, since $\alpha_i^{\max} > 0$ and $\zeta_i > 0$. Thus, we only need to consider the case when $U_i(n) \leq \alpha_i(n+1)$. In this case, we have

$$\begin{aligned}
& (U_i(n)+\zeta_i)\big(\Lambda_i(n+1)-\alpha_i(n+1)\wedge U_i(n)\big) && \text{(B.58)}\\
= \;& (U_i(n)+\zeta_i)\big(\Lambda_i(n+1)-U_i(n)\big) && \text{(B.59)}\\
= \;& (U_i(n)+\zeta_i)\big(\Lambda_i(n+1)-\alpha_i(n+1)\big)+(U_i(n)+\zeta_i)\big(\alpha_i(n+1)-U_i(n)\big) && \text{(B.60)}\\
\leq \;& (U_i(n)+\zeta_i)\big(\Lambda_i(n+1)-\alpha_i(n+1)\big)+(\alpha_i^{\max}+\zeta_i)\alpha_i^{\max}. && \text{(B.61)}
\end{aligned}$$

Thus, the lemma follows from the definition of $L(n)$ in (3.39). ■

We next generalize the above bound from a single time slot to one frame with N time slots.

Lemma B.5.2. *The N-slot drift of $L(n)$ satisfy the following for any control action profile $\{\alpha(n)\}$:*

$$\Delta_N L(n) \leq \sum_{i\in\mathcal{V}}(U_i(n)+\zeta_i)\sum_{\tau=1}^{N}\big(\Lambda_i(n+\tau)-\alpha_i(n+\tau)\big)+N\kappa_1+N^2\kappa_2 + \beta\sum_{\tau=1}^{N}\sum_{j\in\mathcal{J}} f_j(\alpha_{\mathcal{N}_j}(n+\tau);s_{\mathcal{N}_j}(n+\tau))+N\sum_{i\in\mathcal{V}}\alpha_i^{\max}\zeta_i, \quad \text{(B.62)}$$

where κ_1 and κ_2 are sufficiently large constants.

Proof: We carry out the drift analysis for a user $i\in\mathcal{V}$ in Lemma B.5.1 to N time slots and obtain the following:

$$\frac{1}{2}(U_i(n+N)+\zeta_i)^2-\frac{1}{2}(U_i(n)+\zeta_i)^2 \quad \text{(B.63)}$$

$$= \sum_{\tau=1}^{N}\Big(\frac{1}{2}(U_i(n+\tau)+\zeta_i)^2-\frac{1}{2}(U_i(n+\tau-1)+\zeta_i)^2\Big) \quad \text{(B.64)}$$

$$\overset{(a)}{\leq} \sum_{\tau=1}^{N}(U_i(n+\tau-1)+\zeta_i)\big(\Lambda_i(n+\tau)-\alpha_i(n+\tau)\big) \quad \text{(B.65)}$$

$$+N\alpha_i^{\max}\zeta_i+\frac{N}{2}(\Lambda_i^{\max}+\alpha_i^{\max})^2+N(\alpha_i^{\max})^2 \quad \text{(B.66)}$$

$$\overset{(b)}{\leq} \sum_{\tau=1}^{N}\Big((U_i(n)+\zeta_i)\big(\Lambda_i(n+\tau)-\alpha_i(n+\tau)\big)+(\tau-1)\Lambda_i^{\max}(\Lambda_i^{\max}+\alpha_i^{\max})\Big)$$

$$+N\alpha_i^{\max}\zeta_i+\frac{N}{2}(\Lambda_i^{\max}+\alpha_i^{\max})^2+N(\alpha_i^{\max})^2 \quad \text{(B.67)}$$

$$= (U_i(n)+\zeta_i)\sum_{\tau=1}^{N}\big(\Lambda_i(n+\tau)-\alpha_i(n+\tau)\big)+\frac{N(N-1)}{2}\Lambda_i^{\max}(\Lambda_i^{\max}+\alpha_i^{\max})$$

$$+N\alpha_i^{\max}\zeta_i+\frac{N}{2}(\Lambda_i^{\max}+\alpha_i^{\max})^2+N(\alpha_i^{\max})^2, \quad \text{(B.68)}$$

where (a) follows from the bound in Lemma B.5.1, and (b) is because

$$U_i(n+\tau)\leq U_i(n)+\tau\Lambda_i^{\max}. \quad \text{(B.69)}$$

Therefore, the lemma follows. ■

Note that the above bound holds for any control action profile $\{\alpha_i(n)\}$. We next analyze the specific drift of $L^\star(n)$, which is the Lyapunov function under the optimal scheduling policy for

SCH-N. We have the following lemma:

Lemma B.5.3. *The N-slot drift under the solution of SCH-N for each frame m can be bounded as*

$$\Delta_N L^\star(n_m) \leq -\epsilon \sum_{\tau=1}^{N} \sum_{i\in\mathcal{V}} (U_i(n_m+\tau-1)+\zeta_i) + N\sum_{i\in\mathcal{V}} \alpha_i^{\max}\zeta_i + \beta N f_m^\star + N\kappa_3 + N^2\kappa_4,$$

where $n_m = (m-1)N$, and κ_3 and κ_4 are sufficiently large constants.

Proof: We apply the solution to SCH-N to (B.62) and obtain

$$\Delta_N L^\star(n_m) \leq -\epsilon N \sum_{i\in\mathcal{V}} (U_i(n_m)+\zeta_i) + \beta N f_m^\star + N\sum_{i\in\mathcal{V}} \alpha_i^{\max}\zeta_i + N\kappa_1 + N^2\kappa_2, \quad \text{(B.70)}$$

where the term $\beta N f_m^\star$ is due to the fact that $\{\alpha_i(n)\}$ is the optimal control policy. Now, note that

$$U_i(n_m+\tau) \leq U_i(n_m) + \tau\Lambda_i^{\max}. \quad \text{(B.71)}$$

We have

$$\begin{aligned}
\Delta_N L^\star(n_m) &\leq -\epsilon \sum_{\tau=1}^{N}\sum_{i\in\mathcal{V}} (U_i(n_m+\tau)+\zeta_i-\tau\Lambda_i^{\max}) + \beta N f_m^\star \\
&\quad + N\sum_{i\in\mathcal{V}} \alpha_i^{\max}\zeta_i + N\kappa_1 + N^2\kappa_2 \quad \text{(B.72)} \\
&\leq -\epsilon \sum_{\tau=1}^{N}\sum_{i\in\mathcal{V}} (U_i(n_m+\tau)+\zeta_i) + \epsilon\frac{N(N+1)}{2}\sum_{i\in\mathcal{V}}\Lambda_i^{\max} + \beta N f_k^\star \\
&\quad + N\sum_{i\in\mathcal{V}} \alpha_i^{\max}\zeta_i + N\kappa_1 + N^2\kappa_2, \quad \text{(B.73)}
\end{aligned}$$

from which the lemma holds. ■

We are now ready to prove Lemma 3.2.1.

Proof of Lemma 3.2.1: We first compute the N-slot drift with $\{\alpha(n)\}$ computed by Algorithm

3.2.1, as follows:

$$\begin{aligned}
\Delta_N L(n_m) &\leq \sum_{\tau=1}^{N}\sum_{i\in\mathcal{V}}(U_i(n_m+\tau-1)+\zeta_i)(\Lambda_i(n_m+\tau)-\alpha_i(n_m+\tau)) \\
&\quad + \beta\sum_{\tau=1}^{N}\sum_{j\in\mathcal{J}} f_j(\alpha_{\mathcal{N}_j}(n_m+\tau); s_{\mathcal{N}_j}(n_m+\tau)) \\
&\quad + N\sum_{i\in\mathcal{V}}\alpha_i^{\max}\zeta_i + \frac{N}{2}\sum_{i\in\mathcal{V}}(\Lambda_i^{\max}+\alpha_i^{\max})^2 + N\sum_{i\in\mathcal{V}}(\alpha_i^{\max})^2 \\
&\overset{(a)}{\leq} \sum_{\tau=1}^{N}\sum_{i\in\mathcal{V}}(U_i(n_m+\tau-1)+\zeta_i)(\Lambda_i(n_m+\tau)-\alpha_i^\star(n_m+\tau)) \\
&\quad + \beta\sum_{\tau=1}^{N}\sum_{j\in\mathcal{J}} f_j(\alpha^\star_{\mathcal{N}_j}(n_m+\tau); s_{\mathcal{N}_j}(n_m+\tau)) \\
&\quad + N\sum_{i\in\mathcal{V}}\alpha_i^{\max}\zeta_i + \frac{N}{2}\sum_{i\in\mathcal{V}}(\Lambda_i^{\max}+\alpha_i^{\max})^2 + N\sum_{i\in\mathcal{V}}(\alpha_i^{\max})^2 \qquad \text{(B.74)} \\
&\overset{(b)}{\leq} \sum_{i\in\mathcal{V}}(U_i(n_m)+\zeta_i)\sum_{\tau=1}^{N}(\Lambda_i(n_m+\tau)-\alpha_i^\star(n_m+\tau)) \\
&\quad + N\sum_{i\in\mathcal{V}}\alpha_i^{\max}\zeta_i + \beta N f_m^\star + \kappa_5 N + \kappa_6 N^2 \qquad \text{(B.75)} \\
&\overset{(c)}{\leq} -\epsilon N\sum_{i\in\mathcal{V}}(U_i(n_m)+\zeta_i) + \beta N f_m^\star + N\sum_{i\in\mathcal{V}}\alpha_i^{\max}\zeta_i + \kappa_5 N + \kappa_6 N^2 \qquad \text{(B.76)} \\
&\overset{(d)}{\leq} -\epsilon\sum_{i\in\mathcal{V}}\sum_{\tau=1}^{N}(U_i(n_m+\tau-1)+\zeta_i) + \beta N f_m^\star \\
&\quad + N\sum_{i\in\mathcal{V}}\alpha_i^{\max}\zeta_i + B_1 N + B_2 N^2. \qquad \text{(B.77)}
\end{aligned}$$

(a) is because the control action $\alpha(n+1)$ is the solution to the optimization in (3.29), (b) and (d) are obtained by applying the following:

$$U_i(n_m) - \alpha_i^{\max}\tau \leq U_i(n_m+\tau) \leq U_i(n_m) + \tau\Lambda_i^{\max}, \qquad \text{(B.78)}$$

and (c) is because $\{\alpha_i^\star(n)\}$ solves SCH-N. ■

APPENDIX C

PROOFS IN CHAPTER 4

C.1 PROOF OF LEMMA 4.1.1

Proof: We can write the equality constraints in SCH-L as follows:

$$\begin{pmatrix} A & \lambda \\ 1^T & 0 \end{pmatrix} \begin{pmatrix} x \\ \gamma \end{pmatrix} = \begin{pmatrix} \lambda \\ 1 \end{pmatrix}. \tag{C.1}$$

It is not difficult to verify that the initial vertex as described by (4.6) and (4.7) is feasible. Thus, the changes in the variables $(\Delta x^T, \Delta\gamma)^T$ should always lie in the null space of the matrix $\begin{pmatrix} A & \lambda \\ 1^T & 0 \end{pmatrix}$. Given the new column α_{new}, this implies that the existing coefficients should satisfy the following:

$$\begin{pmatrix} B & \lambda \\ 1^T & 0 \end{pmatrix} \begin{pmatrix} \Delta y \\ \Delta\gamma \end{pmatrix} + \Delta z \begin{pmatrix} \alpha_{\text{new}} \\ 1 \end{pmatrix} = 0. \tag{C.2}$$

where $\Delta z \geq 0$ is the change of the scheduling variable associated with the new column α_{new}. From the first equality in (C.2), we have

$$\lambda\Delta\gamma = -\Delta z\alpha_{\text{new}} - B\Delta y. \tag{C.3}$$

Multiplying both sides with $1^T B^{-1}$ and noting that $1^T \Delta y = -\Delta z$, we have

$$\begin{aligned} \Delta\gamma &= \frac{\Delta z}{1^T B^{-1}\lambda}(1 - 1^T B^{-1}\alpha_{\text{new}}) && \text{(C.4)} \\ &\overset{(a)}{\leq} \frac{\Delta z}{1^T B^{-1}\lambda}(1 - 1^T B^{-1}\lambda), && \text{(C.5)} \end{aligned}$$

where (a) is because α_{new} is by maximizing the function in (4.8), and thereby satisfies

$$1^T B^{-1}\alpha_{\text{new}} \geq 1^T B^{-1}\lambda, \tag{C.6}$$

since λ is a convex combination of the columns of A. We now show that $\Delta\gamma \leq 0$. Note that from (4.12) we have

$$y = (1-\gamma)B^{-1}\lambda. \tag{C.7}$$

Multiplying both sides of the above by 1^T and noting that $1^T y = 1$, we obtain

$$1^T B^{-1}\lambda = \frac{1}{1-\gamma} \tag{C.8}$$

Thus, we can write the last inequality in (C.5) as follows:

$$\begin{aligned} \Delta\gamma &\leq \Delta z(1-\gamma)(1 - \frac{1}{1-\gamma}) && \text{(C.9)} \\ &= -\gamma\Delta z. && \text{(C.10)} \end{aligned}$$

Thus, we conclude that α_{new} is a cost decreasing direction, and the lemma holds. ■

C.2 PROOF OF LEMMA 4.1.2

Proof: Suppose $z_{\text{new}} = 0$, then the solution to (4.9) is the same as the one associated with the old vertex as specified by matrix B. This contradicts with the fact that α_{new} is a cost-decreasing direction, as proved in Lemma 4.1.1. Now, assume that $z_{\text{new}} > 0$. Note that the cost reduction is at least proportional to z_{new}, according to (C.10). Since the objective function is bounded below, we conclude that z_{new} is finite. This only happens if some coefficient in y reaches zero for the first time, so that certain inequality constraint in (4.9) becomes active. Therefore, the lemma holds. ■

C.3 PROOF OF LEMMA 4.2.1

Proof: We first form the Lagrangian of (4.9) as follows:

$$\begin{aligned} \text{minimize}_{\{y,z,\gamma\}} \quad & \gamma + \theta^T((1-\gamma)\hat{\lambda} - By - \alpha_{\text{new}} z) \\ \text{subject to} \quad & 1^T y + z = 1 \\ & y \succeq 0, z \geq 0. \end{aligned} \tag{C.11}$$

Note that this is a linear programming problem where the variables $(y^T, z)^T$ lie in a simplex. We further write the objective function as follows

$$f(y, z, \gamma) = (1 - \theta^T \hat{\lambda})\gamma - \theta^T(By + \alpha_{\text{new}} z) + \theta^T \hat{\lambda}. \tag{C.12}$$

Thus, given fixed $\theta(n)$, the optimal primal variable in (y, z) can be obtained by choosing the column in B or α_{new} which has the largest weight. This is is implemented in (4.21). Now, consider the static problem (4.9) as a convex optimization problem in variable γ only. It is not difficult to see that $\left(1 - \theta(n)^T \hat{\lambda}\right)$ is the sub-gradient of γ. Further, notice that $\left((1-\gamma(n))\hat{\lambda} - \alpha(n)\right)$ is a sub-gradient of $\theta(n)$, according to (C.11). Therefore, we conclude that the algorithm is a standard sub-gradient method for a convex optimization problem from which the convergence result holds. ■

C.4 PROOF OF LEMMA 4.2.2

Proof: For notation simplicity, we assume that one column in B is already replaced with α_{new}, and that the coefficients are relabeled accordingly. We only need to show that at the the convergence of $\{\theta(n)\}$, we have

$$\theta(n)^T B = (1-\gamma)1^T. \tag{C.13}$$

Thus, at the convergence, all the schedules in B have equal weights. Assuming (C.13), we then have

$$\theta(n)^T \alpha = (1-\gamma)1^T B^{-1} \alpha, \tag{C.14}$$

from which the lemma holds.

Now we prove (C.13) as follows. Notice that the Lagrangian for (4.9) is

$$f(y,\gamma) = (1-\theta^T\lambda)\gamma - \theta^T B y + \theta^T\lambda. \tag{C.15}$$

Thus, if convergence is achieved, all columns in B should have the same weight, since only the column with the maximum weight will have nonzero coefficient for the optimal solution, due to the fact that the scheduling variables y lie in a simplex. Further, note that from (C.12) we have

$$\theta(n)^T\lambda = 1, \tag{C.16}$$

and the feasibility of (4.9) implies that

$$(1-\gamma)\lambda = By. \tag{C.17}$$

Thus, we conclude that

$$\begin{aligned} 1 &= \theta(n)^T\lambda && \text{(C.18)}\\ &= \frac{1}{1-\gamma}\theta(n)^T B y && \text{(C.19)}\\ &= \frac{c_0}{1-\gamma}1^T y && \text{(C.20)}\\ &\overset{(a)}{=} \frac{c_0}{1-\gamma}, && \text{(C.21)} \end{aligned}$$

where (a) is because y lies in a simplex. Therefore, the lemma holds. ■

C.5 PROOF OF LEMMA 4.2.3

Proof: Since $\hat{\lambda}$ is not changing and that the schedules B and α_{new} can achieve a non-positive throughput gap, from Algorithm 4.2.1 we conclude that the initialization of basic matrix B in step 2 is never executed. Further, the column generation step in (4.24) is never executed. Notice that from

(4.22) we have

$$\frac{1}{2r_n\delta\epsilon}(\theta_i(r_n(t_0+\delta))-\theta_i(r_n(t_0-\delta)))=(1-\frac{1}{2r_n\delta}\sum_{n=r_n(t_0-\delta)}^{r_n(t_0+\delta)}\gamma(\tau))\hat{\lambda}_i-\frac{1}{2r_n\delta}\sum_{\tau=r_n(t_0-\delta)}^{r_n(t_0+\delta)}\alpha_i(n).$$

Since the sequence $\{\theta(n)\}$ is bounded, we conclude that

$$\begin{aligned} 0 &= \lim_{n\to\infty}\frac{1}{2r_n\delta\epsilon}(\theta_i(r_n(t_0+\delta))-\theta_i(r_n(t_0-\delta))) && \text{(C.22)}\\ &= \lim_{n\to\infty}\Big\{(1-\frac{1}{2r_n\delta}\sum_{\tau=r_n(t_0-\delta)}^{r_n(t_0+\delta)}\gamma(\tau))\hat{\lambda}_i-\frac{1}{2r_n\delta}\sum_{\tau=r_n(t_0-\delta)}^{r_n(t_0+\delta)}\alpha_i(n)\Big\}, && \text{(C.23)}\end{aligned}$$

from which the lemma holds. ■

APPENDIX **D**

PROOFS IN CHAPTER 5

D.1 PROOF OF LEMMA 5.2.1

Proof: It is sufficient to prove that $\{\Delta_{ij}\}$ belongs to the set $\mathcal{W}$. Note that for the set of weights $\{\Delta_{ij}\}$ defined in (5.20), we have $\Delta_{ij} \geq 0$, $\Delta_{ii} = 0$, and $\Delta_{ij} = \Delta_{ji}$ for all $i, j \in \mathcal{V}$. Further, $\Delta_{ij} = 0$ if i and j are not neighbors. Now, for any factor node k that includes user i and any maximal schedule α such that $\alpha_i = 0$, we have

$$\sum_{j \in \mathcal{N}_k} \Delta_{ij}\alpha_j \overset{(a)}{\geq} \sum_{j \in \mathcal{N}_k} \frac{\alpha_j \mathbf{1}_{\{\alpha_j > 0\}}}{\alpha_j^{\min}} \nu_{ij} \tag{D.1}$$

$$\geq \sum_{j \in \mathcal{N}_k} \nu_{ij} \mathbf{1}_{\{\alpha_j > 0\}} \tag{D.2}$$

$$\overset{(b)}{\geq} 1, \tag{D.3}$$

where (a) is because of the definition in (5.20), and (b) is because of (5.21) and the fact that α is maximal. Thus, we conclude that the matrix $\{\Delta_{ij}\}$ belongs to $\mathcal{W}$, and therefore, the lemma holds according to Theorem 5.2.1. ■

D.2 PROOF OF LEMMA 5.2.2

Proof: Let a user $i \in \mathcal{V}$ be given. Since $\lambda \in \mathcal{R}^\star$, we assume that there is a scheduler π such that $\lambda \in \mathcal{R}_\pi$. For any such scheduler π, consider the following 'Lyapunov' function

$$L_i(n) = \frac{1}{\alpha_i^{\min}} U_i(n) + \sum_{j \in \mathcal{N}_i} \Delta_{ij} U_j(n) \tag{D.4}$$

$$= \frac{1}{\alpha_i^{\min}} \Big(U_i(0) + \Lambda_i(n) - D_i(n) \Big) + \sum_{j \in \mathcal{N}_i} \Delta_{ij} (U_j(0) + \Lambda_j(n) - D_j(n)). \tag{D.5}$$

Since the network is rate stable under the scheduler π, we have

$$\lim_{n \to \infty} \frac{L_i(n)}{n} = 0, \text{ w.p.1}, \tag{D.6}$$

which implies that w.p.1,

$$\lim_{n \to \infty} \frac{\frac{1}{\alpha_i^{\min}} \Lambda_i(n) + \sum_{j \in \mathcal{N}_i} \Delta_{ij} \Lambda_j(n)}{n} = \lim_{n \to \infty} \frac{\frac{1}{\alpha_i^{\min}} D_i(n) + \sum_{j \in \mathcal{N}_i} \Delta_{ij} D_j(n)}{n}$$

$$\overset{(a)}{\leq} \Delta_i \tag{D.7}$$

$$\leq \Delta, \tag{D.8}$$

where (a) is because of the upper bound on the total departures in each time slot in (5.25). Therefore, the lemma follows from the SLLN on the arrival processes and the fact that the scheduler π is chosen arbitrarily. ■

D.3 PROOF OF LEMMA 5.3.1

Proof: According to the scheduler, the links in $\mathcal{N}_i^p$ are always considered before link i. Thus, when a back-logged link i is being considered by the scheduler, either there is already a scheduled link in $\mathcal{N}_i^p$, or link i is put to the schedule. In both cases, there is at least one packet departure among the links in $\{i\} \cup \mathcal{N}_i^p$, from which the lemma follows. ■

REFERENCES

[1] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: The next computing revolution," in *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, Jun. 2010, pp. 731–736.

[2] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*, 1st ed. Lee and Seshia, 2010. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/794.html.

[3] K.-D. Kim and P. Kumar, "Cyber physical systems: A perspective at the centennial," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1287–1308, 13 2012.

[4] S. Karnouskos, "Cyber-physical systems in the smartgrid," in *Proceedings of the IEEE International Conference on Industrial Informatics (INDIN)*, Jul. 2011, pp. 20–23.

[5] L. Parolini, B. Sinopoli, B. Krogh, and Z. Wang, "A cyber physical systems approach to data center modeling and control for energy efficiency," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 254–268, Jan. 2012.

[6] L. Rao, X. Liu, M. Ilic, and J. Liu, "Distributed coordination of internet data centers under multiregional electricity markets," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 269–282, Jan. 2012.

[7] (2008, Nov.) National workshop on research on transportation cyber-physical systems: Automotive, aviation, and rail. [Online]. Available: http://www.ee.washington.edu/research/nsl/aar-cps.

[8] I. Lee, O. Sokolsky, S. Chen, J. Hatcliff, E. Jee, B. Kim, A. King, M. Mullen-Fortino, S. Park, A. Roederer, and K. Venkatasubramanian, "Challenges and research directions in medical cyber physical systems," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 75–90, Jan. 2012.

[9] N. Abramson, "The ALOHA system: another alternative for computer communications," in *Proceedings of the November 17-19, 1970, fall joint computer conference*, ser. AFIPS '70 (Fall). New York, NY, USA: ACM, 1970, pp. 281–285.

[10] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 910–917, Sep. 1988.

[11] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936 –1948, Dec. 1992.

[12] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, vol. 2, Mar./Apr. 1998, pp. 533–539 vol.2.

[13] E. Modiano, D. Shah, and G. Zussman, "Maximizing throughput in wireless networks via gossiping," in *Proceedings of the joint international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 2006, pp. 27–38.

[14] P. Chaporkar, K. Kar, L. Xiang, and S. Sarkar, "Throughput and fairness guarantees through maximal scheduling in wireless networks," *IEEE Transactions on Information Theory*, vol. 54, no. 2, pp. 572–594, Feb. 2008.

[15] L. Jiang and J. Walrand, "A distributed csma algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 3, pp. 960 –972, Jun. 2010.

[16] J. Ni, B. Tan, and R. Srikant, "Q-csma: Queue-length-based csma/ca algorithms for achieving maximum throughput and low delay in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, pp. 825–836, Jun. 2012.

[17] G. Kim, Q. Li, and R. Negi, "A graph-based algorithm for scheduling with sum-interference in wireless networks," in *Proceedings of the IEEE Global Telecommunications Conference*, Nov. 2007, pp. 5059–5063.

[18] ——, "A polynomial-time approximation algorithm for weighted sum-rate maximization in UWB networks," in *Proceedings of the IEEE International Conference on Communications*, May 2008, pp. 3775–3779.

[19] Q. Li, G. Kim, and R. Negi, "Maximal scheduling in a hypergraph model for wireless networks," in *Proceedings of the IEEE International Conference on Communications*, May 2008, pp. 3853–3857.

[20] Q. Li and R. Negi, "Prioritized maximal scheduling in wireless networks," in *Proceedings of the IEEE Global Telecommunications Conference*, Dec. 2008, pp. 1–5.

[21] ——, "Back-pressure routing and optimal scheduling in wireless broadcast networks," in *Proceedings of the IEEE Global Telecommunications Conference*, Dec. 2009, pp. 1–6.

[22] ——, "Scheduling in multi-hop wireless networks with priorities," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, Apr. 2009, pp. 2926–2930.

[23] ——, "Scheduling in wireless networks under uncertainties," in *Proceedings of the IEEE International Conference on Communications*, May 2010, pp. 1–5.

[24] ——, "Greedy maximal scheduling in wireless networks," in *Proceedings of the IEEE Global Telecommunications Conference*, Dec. 2010, pp. 1–5.

[25] ——, "Scheduling in wireless networks under uncertainties: A greedy primal-dual approach," in *Proceedings of the IEEE International Conference on Communications*, Jun. 2011, pp. 1–5.

[26] ——, "Maximal scheduling in wireless ad hoc networks with hypergraph interference models," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 1, pp. 297–310, Jan. 2012.

[27] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: a media access protocol for wireless LAN's," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 4, pp. 212–225, Oct. 1994.

[28] A. Boulanger, A. Chu, S. Maxx, and D. Waltz, "Vehicle electrification: Status and issues," *Proceedings of the IEEE*, vol. 99, no. 6, pp. 1116–1138, Jun. 2011.

[29] K. Clement-Nyns, E. Haesen, and J. Driesen, "The impact of charging plug-in hybrid electric vehicles on a residential distribution grid," *IEEE Transactions on Power System*, vol. 25, no. 1, pp. 371–380, Feb. 2010.

[30] J. Lopes, F. Soares, and P. Almeida, "Integration of electric vehicles in the electric power system," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 168–183, Jan. 2011.

[31] L. Gan, U. Topcu, and S. Low, "Optimal decentralized protocols for electric vehicle charging," in *Proceedings of the IEEE Conference on Decision and Control*, 2011.

[32] Z. Ma, D. Callaway, and I. Hiskens, "Decentralized charging control for large populations of plug-in electric vehicles: Application of the nash certainty equivalence principle," in *Proceedings of the IEEE Conference on Decision and Control*, Sep. 2010, pp. 191–195.

[33] E. Sortomme, M. Hindi, S. MacPherson, and S. Venkata, "Coordinated charging of plug-in hybrid electric vehicles to minimize distribution system losses," *IEEE Transactions on Smart Grid*, vol. 2, no. 1, pp. 198–205, Mar. 2011.

[34] N. Rotering and M. Ilic, "Optimal charge control of plug-in hybrid electric vehicles in deregulated electricity markets," *IEEE Transactions on Power System*, vol. 26, no. 3, pp. 1021–1029, Aug. 2011.

[35] O. Sundstroem and C. Binding, "Optimization methods to plan the charging of electric vehicle fleets," in *Proceedings of the Intnational Conference on Control, Communication and Power Engineering*, 2010.

[36] K. Turitsyn, N. Sinitsyn, S. Backhaus, and M. Chertkov, "Robust broadcast-communication control of electric vehicle charging," in *Proceedings of the IEEE SmartGridComm*, Oct. 2010, pp. 203–207.

[37] Q. Li and R. Negi, "Distributed scheduling in cyber-physical systems: The case of coordinated electric vehicle charging," in *Proceedings of the IEEE Global Telecommunications Conference Workshops*, Dec. 2011, pp. 1183–1187.

[38] K. Schneider, C. Gerkensmeyer, M. Kintner-Meyer, and R. Fletcher, "Impact assessment of plug-in hybrid vehicles on pacific northwest distribution systems," in *Proceedings of the IEEE Power Engineering Society General Meeting*, Jul. 2008, pp. 1–6.

[39] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making scheduling "cool": temperature-aware workload placement in data centers," in *Proceedings of the annual conference on USENIX Annual Technical Conference*, ser. ATEC '05. Berkeley, CA, USA: USENIX Association, 2005, pp. 5–5.

[40] Q. Tang, S. Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 11, pp. 1458–1472, Nov. 2008.

[41] L. Parolini, N. Tolia, B. Sinopoli, and B. H. Krogh, "A cyber-physical systems approach to energy management in data centers," in *Proceedings of the 1st ACM/IEEE International Con-*

ference on Cyber-Physical Systems, ser. ICCPS '10. New York, NY, USA: ACM, 2010, pp. 168–177.

[42] E. Pakbaznia, M. Ghasemazar, and M. Pedram, "Temperature-aware dynamic resource provisioning in a power-optimized datacenter," in *Proceedings of the Conference on Design, Automation and Test in Europe*. 3001 Leuven, Belgium, Belgium: European Design and Automation Association, 2010, pp. 124–129.

[43] R. Urgaonkar, U. Kozat, K. Igarashi, and M. Neely, "Dynamic resource allocation and power management in virtualized data centers," in *Proceedings of the IEEE Network Operations and Management Symposium (NOMS)*, Apr. 2010, pp. 479–486.

[44] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M. Neely, "Data centers power reduction: A two time scale approach for delay tolerant workloads," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, Mar. 2012, pp. 1431–1439.

[45] J. Moore, R. Sharma, R. Shih, J. Chase, R. Patel, and P. Ranganathan, "Going beyond CPUs: The potential of temperature-aware solutions for the data center," in *Proceedings of the Workshop of Temperature-Aware Computer Systems (TACS-1) held at ISCA*, 2004.

[46] H. Luo, S. Lu, and V. Bharghavan, "A new model for packet scheduling in multihop wireless networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, ser. MobiCom '00. New York, NY, USA: ACM, 2000, pp. 76–86.

[47] X. Lin and N. Shroff, "The impact of imperfect scheduling on cross-layer congestion control in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 302–315, Apr. 2006.

[48] X. Wu, R. Srikant, and J. Perkins, "Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 6, pp. 595–605, Jun. 2007.

[49] G. Sharma, R. R. Mazumdar, and N. B. Shroff, "On the complexity of scheduling in wireless networks," in *Proceedings of the 12th annual international conference on Mobile computing and networking*, ser. MobiCom '06. New York, NY, USA: ACM, 2006, pp. 227–238.

[50] A. Hasan and J. Andrews, "The guard zone in wireless ad hoc networks," *IEEE Transactions on Wireless Communications*, vol. 6, no. 3, pp. 897–906, Mar. 2007.

[51] G. Krumpholz, K. Clements, and P. Davis, "Power system observability: A practical algorithm using network topology," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-99, no. 4, pp. 1534–1542, Jul. 1980.

[52] K. Clements, G. Krumpholz, and P. Davis, "Power system state estimation residual analysis: An algorithm using network topology," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-100, no. 4, pp. 1779 –1787, Apr. 1981.

[53] O. Kosut, L. Jia, R. Thomas, and L. Tong, "Malicious data attacks on the smart grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 645 –658, Dec. 2011.

[54] J. Lavaei and S. Low, "Zero duality gap in optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 92–107, Feb. 2012.

[55] B. Zhang and D. Tse, "Geometry of feasible injection region of power networks," in *Proceedings of the 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sep. 2011, pp. 1508–1515.

[56] M. Kraning, E. Chu, J. Lavaei, and S. Boyd, "Message passing for dynamic network energy management," Stanford University, Tech. Rep., Apr. 2012. [Online]. Available: http://www.stanford.edu/∼boyd/papers/decen_dyn_opt.html.

[57] M. Neely, "Energy optimal control for time-varying wireless networks," *IEEE Transactions on Information Theory*, vol. 52, no. 7, pp. 2915–2934, Jul. 2006.

[58] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, pp. 1333–1344, Dec. 2007.

[59] D. Shah, J. Shin, and P. Tetali, "Medium access using queues," in *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, ser. FOCS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 698–707.

[60] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends in Machine Learning*, vol. 1, no. 1-2, pp. 1–305, Jan. 2008.

[61] W. Su and M.-Y. Chow, "Performance evaluation of an EDA-based large-scale plug-in hybrid electric vehicle charging algorithm," *IEEE Transactions on Smart Grid*, vol. 3, no. 1, pp. 308–315, Mar. 2012.

[62] P. Richardson, D. Flynn, and A. Keane, "Optimal charging of electric vehicles in low-voltage distribution systems," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 268–279, Feb. 2012.

[63] S. Kompella, J. Wieselthier, A. Ephremides, H. Sherali, and G. Nguyen, "On optimal SINR-based scheduling in multihop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 6, pp. 1713 –1724, Dec. 2010.

[64] D. Shah, D. Tse, and J. Tsitsiklis, "Hardness of low delay network scheduling," *IEEE Transactions on Information Theory*, to appear.

[65] L. Jiang, D. Shah, J. Shin, and J. Walrand, "Distributed random access algorithm: scheduling and congestion control," *IEEE Transactions on Information Theory*, vol. 56, no. 12, pp. 6182–6207, Dec. 2010.

[66] X. Lin and S. Rasool, "Constant-time distributed scheduling policies for ad hoc wireless networks," *IEEE Transactions on Automatic Control*, vol. 54, no. 2, pp. 231–242, Feb. 2009.

[67] A. Gupta, X. Lin, and R. Srikant, "Low-complexity distributed scheduling algorithms for wireless networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 6, pp. 1846–1859, Dec. 2009.

[68] A. Dimakis and J. Walrand, "Sufficient conditions for stability of longest-queue-first scheduling: Second-order properties using fluid limits," *Advances in Applied Probability*, vol. 38, no. 2, pp. pp. 505–521, 2006.

[69] C. Joo, X. Lin, and N. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multihop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1132–1145, Aug. 2009.

[70] G. Zussman, A. Brzezinski, and E. Modiano, "Multihop local pooling for distributed throughput maximization in wireless networks," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, Apr. 2008, pp. 1139–1147.

[71] M. Leconte, J. Ni, and R. Srikant, "Improved bounds on the throughput efficiency of greedy maximal scheduling in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 3, pp. 709–720, Jun. 2011.

[72] B. Birand, M. Chudnovsky, B. Ries, P. Seymour, G. Zussman, and Y. Zwols, "Analyzing the performance of greedy maximal scheduling via local pooling and graph theory," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 163–176, Feb. 2012.

[73] C. Joo and N. Shroff, "Local greedy approximation for scheduling in multihop wireless networks," *IEEE Transactions on Mobile Computing*, vol. 11, no. 3, pp. 414–426, Mar. 2012.

[74] J. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, vol. 2, 2000, pp. 556–564.

[75] J. G. Dai, "On positive Harris recurrence of multiclass queueing networks: A unified approach via fluid limit models," *Annals of Applied Probability*, vol. 5, pp. 49–77, 1995.

[76] A.L.Stolyar, "On the stability of multiclass queueing networks: A relaxed sufficient condition via limiting fluid processes," *Markov Processes and Related Fields*, pp. 491–512, 1995.

[77] S. Sarkar and K. Sivarajan, "Hypergraph models for cellular mobile communication systems," *IEEE Transactions on Vehicular Technology*, vol. 47, no. 2, pp. 460–471, May 1998.

[78] O. Goussevskaia, Y.-A. Pignolet, and R. Wattenhofer, "Efficiency of wireless networks: Approximation algorithms for the physical interference," *Foundations and Trends in Networking*, vol. 4, no. 3, pp. 313–420, 2010.

[79] N. Bansal, T. Kimbrel, and K. Pruhs, "Speed scaling to manage energy and temperature," *Journal of the ACM (JACM)*, vol. 54, no. 1, pp. 3:1–3:39, Mar. 2007.

[80] M. Neely, *Stochastic network optimization with application to communication and queueing systems*. Morgan Claypool, 2010.

[81] W. Kersting, "Radial distribution test feeders," in *Proceedings of the IEEE Power Engineering Society Winter Meeting*, vol. 2, 2001, pp. 908–912.

[82] National Renewable Energy Labratory. (2006) Wind integration data sets. [Online]. Available: http://www.nrel.gov/wind/integrationdatasets/eastern/data.html.

[83] Southern California Edision. (2011) Regulatory information - SCE load profiles. [Online]. Available: http://www.sce.com/AboutSCE/Regulatory/loadprofiles.

[84] P. Hu and T. Reuscher. (2004, Dec.) Summary of travel trends. U.S. Department of Transportation and Federal Highway Administration. [Online]. Available: http://nhts.ornl.gov/2001/pub/STT.pdf.

[85] C. Zillober, K. Schittkowski, and K. Moritzen, "Very large scale optimization by sequential convex programming," *Optimization Methods and Software*, vol. 19, no. 1, pp. 103–120, 2004.

[86] D. Bertsimas and J. Tsitsiklis, *Introduction to Linear Optimization*, 1st ed. Athena Scientific, 1997.

[87] E. Arikan, "Some complexity results about packet radio networks," *IEEE Transactions on Information Theory*, vol. 30, no. 4, pp. 681–685, Jul. 1984.

[88] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed. Athena Scientific, 2000.

[89] M. Morari, C. Garcia, and D. M. Prett, "Model predictive control: Theory and practice – A survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[90] D. Stoyan, W. Kendall, and J. Mecke, *Stochastic geometry and its applications*. Wiley, 1987.

[91] M. Haenggi and R. K. Ganti, "Interference in large wireless networks," *Foundations and Trends in Networking*, vol. 3, no. 2, pp. 127–248, Feb. 2009.

[92] M. Haenggi, "On distances in uniformly random networks," *IEEE Transactions on Information Theory*, vol. 51, no. 10, pp. 3584 –3586, Oct. 2005.

[93] L. B. Le, E. Modiano, C. Joo, and N. B. Shroff, "Longest-queue-first scheduling under SINR interference model," in *Proceedings of the eleventh ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2010, pp. 41–50.

[94] H. Royden, *Real Analysis*. Prentice Hall, 1988.

[95] P. Billingsley, *Probability and Measure*. Wiley, 1979.