Complexity in Wireless Scheduling: Impact and Tradeoffs*

Yung Yi
EE Dept. Princeton Univ.
Princeton, NJ, USA
yyi@princeton.edu

Alexandre Proutière
Microsoft Research
Cambirdge, UK
Alexandre.Proutiere
@microsoft.com

Mung Chiang
EE Dept. Princeton Univ.
Princeton, NJ, USA
chiangm@princeton.edu

ABSTRACT

It has been an important research topic since 1992 to maximize stability region in constrained queueing systems, which includes the study of scheduling over wireless ad hoc networks. In this paper, we propose a framework to study a wide range of existing and future scheduling algorithms and characterize the achieved tradeoffs in stability, delay, and complexity. These characterizations reveal interesting properties hidden in the study of any one or two dimensions in isolation. For example, decreasing complexity from exponential to polynomial, while keeping stability region the same, generally comes at the expense of exponential growth of delays. Investigating trade-offs in the 3-dimensional space allows a designer to fix one dimension and vary the other two jointly. For example, incentives for using scheduling algorithms with only partial throughput-guarantee can be quantified with regards to delay and complexity. Trade-off analysis is then extended to systems with congestion control through utility maximization for non-stabilizable arrival inputs, where the complexity-utility-delay trade-off is shown to be different from the complexity-stability-delay tradeoff. Finally, we analyze more practical models with bounded message size, and consider "effective throughput" which reflects resource occupied by control messages. We show that effective throughput may degrade significantly in certain scheduling algorithms, and suggest a mechanism to avoid this problem in light of the 3D tradeoff framework.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Algorithms, Performance, Theory

Keywords

Scheduling, Wireless Networks, Tradeoff, Complexity

*This work has been supported in part by ONR N00014-07-1-0864, NSF CCF-0430487,and CNS-0720570.

MobiHoc'08, May 26–30, 2008, Hong Kong SAR, China.

1. INTRODUCTION

1.1 Overview

Since the seminal work [25] on throughput maximization for constrained queueing systems, there have been growing interests in resource allocation algorithms with provable, topology-independent throughput-guarantees over wireless multi-hop networks. As will be defined in the next section, throughput, measured by stability region, becomes the object to be maximized. To overcome the exponential computational complexity of the optimal algorithm in [25] (referred to as Max-Weight scheduling), a discussion on complexity was initiated and a randomized algorithm was developed in [24]. Distributed scheduling has been extensively studied over the last several years, including Maximal and Greedy scheduling algorithms [1, 12, 21, 23, 27], decentralized Pick-and-Compare algorithms [4, 14, 20] motivated by [24], and constant-time random access algorithms [6, 11].

Then, the authors in [19,20] explicitly brought up another dimension in addition to stability: the dimension of time complexity, and developed *parameterized* algorithms that achieve arbitrary trade-off between complexity and stability[1]. However, stability is an asymptotic concept, and thus its trade-off only with time complexity may not be well-defined. In other words, certain notions of complexity may be arbitrarily reduced without affecting throughput. This raises a question on other performance metrics that has to be paid as a cost for complexity reduction.

In this paper, we aim at proposing and studying a generalized framework which quantifies performance and complexity, and their underlying tradeoffs in wireless scheduling as a global perspective. To that end, we first parameterize a family of scheduling algorithms that include major classes known in existing literature and even new classes. This family is called (γ, ξ, χ)-approximate algorithms, where γ and ξ relate to the achievable throughput and delay, respectively, and χ is the complexity parameter. We use this parameterization to *(i)* explain the root causes of complexity reduction in low-cost (distributed) scheduling algorithms, *(ii)* quantify their impact on throughput and delay performance, and *(iii)* study the stability-delay-complexity trade-off and its engineering implications on the design of scheduling algorithms.

This 3-dimensional trade-off space highlights the following point: an appropriate framework to compare the large set of alternative scheduling algorithms is to view each one as one point in the 3D space on some achievable trade-off

[1]More precisely, trade-off between temporal "simplicity" of scheduling algorithm and the set of stabilizable arrival rates.

surface. The 3-way trade-off relationship resolves potential problems when trading-off two of the dimensions while letting the other dimension "float", e.g., complexity-stability trade-off becomes well-defined for a fixed delay performance. In comparison of two scheduling algorithms, a fair comparison can be made on the other dimensions, by equalizing one dimension. Pictorially, this amounts to determining under what settings of network parameters will one point in the 3D space be Pareto-dominant over the other. We extend our analysis to the systems using congestion control for arrivals outside the stability region. Complexity-utility-delay trade-off is characterized, and now complexity may also negatively impact utility in addition to delay.

Finally, we also consider the practical setup where in distributed algorithms the control message size is bounded, and can include only a limited amount of information. By introducing the notion of effective throughput to explicitly consider time-resource used by control message, we provide more practical analysis of existing queue-length based algorithms, propose a different ways of using queue-length, and quantify its impact on the tradeoff among effective throughput, delay, and complexity.

1.2 Related Work

In addition to the seminal work and low-complexity scheduling algorithms mentioned earlier, the average delay-bound performance of the Max-Weight (MW) scheduling and its variants were studied in switch scheduling [10, 16, 22] and in wireless scheduling [7,17]. The delay-bound analysis is based on the Lyapunov technique, originally introduced in [13] for Markovian systems, which we also use in this paper. In particular, the authors in [7, 16] also studied the tradeoff between complexity and delay by introducing the "periodic computation" of optimal schedules. Periodic computation was also addressed as a source of complexity reduction without loss of throughput in [2]. The work in [2] introduced the notion of "ξ-inaccurate policy" and proved that the Pick-and-Compare scheduling is ξ-inaccurate, but no delay analysis was presented. The case for non-stabilizable input arrivals was discussed in, e.g., [2, 15] with utility maximization subject to system stability or with Aloha-based random access [9]. As Section 3 shows, our definition of (γ, ξ, χ) generalizes ξ-inaccurate policy in [2] with delay analysis. Our work generalizes the complexity-delay tradeoff studied only for the MW scheduling and its periodic computations in [7, 16], and also considers sub-optimal scheduling algorithms as well as utility-delay-complexity in our framework.

While the delay bound based on Lyapunov technique and the tradeoff between complexity and delay have been studied for a subset of scheduling algorithms, what remains to be understood include (i) a generalized, yet tractable framework that can include various optimal/sub-optimal algorithms, and (ii) study of their complexity and its impact on performance under different systems (e.g., with or without congestion control). Under the generalized framework in this paper, we can compare a wide class of scheduling algorithms, recover a large amount of recently obtained results, and potentially predict new performance characterization of scheduling algorithms in the future. In the last part of the paper, we also show that our framework can be extended to study practical systems where message size is bounded and its impact on tradeoffs.

1.3 Main Contributions and Organization

1) In Section 3, we propose a generalized family of scheduling algorithms, called (γ, ξ, χ)-approximate algorithms. This allows us to include major classes scheduling algorithms today, such as Greedy, Locally-Greedy, and Pick-and-Compare algorithms, followed by quantification of throughput and average delay.
2) In Section 4, we use the definition of (γ, χ, ξ)-approximate algorithms to mathematically understand the root causes of complexity reduction in various low-complexity (distributed) scheduling algorithms, and quantify the impacts of complexity reduction on throughput and delay. Based on this, we study all three cases of two-dimensional trade-offs, provided that one dimension is fixed. This provides engineering implications on the choice of scheduling algorithms in practical systems. For example, for a fixed complexity (resp. delays), we understand the conditions under which it is better to choose suboptimal algorithms rather than optimal ones in terms of throughput. In Section 5, we extend our results to the systems with congestion control for non-stabilizable inputs and quantify the utility-delay-complexity tradeoff. We show that a popular method of complexity reduction that does not affect stability in stability-delay-complexity tradeoff now affects the achieved utility.
3) In Section 6, we explore a related aspect of complexity measure, the impact of bounding the size of control messages in distributed scheduling. The measure of "effective throughput" explicitly reflects complexity in light of bounded message size, and is shown to possibly degrade significantly for certain queue-size based scheduling. This motivates us to study a modified version of the algorithms that uses a time-differential queue-length, and to characterize impact of this design on 3D tradeoffs.

2. MODEL AND PERFORMANCE METRICS

2.1 System Model

Network and Traffic Model. The network is represented by a graph $G(\mathcal{L}, \mathcal{V})$, where $\mathcal{L}$ and $\mathcal{V}$ denote the set of directional links, and the set of nodes, respectively. Denote by L and V the numbers of links and nodes. The network handles traffic from a set $\mathcal{F}$ of flows (i.e., sessions), each identified by a set of source and a set of destination nodes. Denote by F the number of sessions. We assume that each node $v \in \mathcal{V}$ maintains a set of per-session queues with finite size. Each source node is fed by exogenous arrivals, and we denote by $A_{f,v}(t)$ the exogenous traffic (in packets/slot) generated by session f during time-slot t at node v.[2] The arrivals $A_{f,v}(t)$ are assumed to be i.i.d. across time-slots, with mean $\mathbb{E}[A_{f,v}(t)] = \lambda_{f,v}$. We assume that the duration of a time-slot is small enough so that $A_{f,v}(t)$ is upper bounded by a constant, $A_{\max}$. Define the arrival vector by $\boldsymbol{\lambda} = (\lambda_{f,v}, f \in \mathcal{F}, v \in \mathcal{V})$.

Network resource and allocation scheme. The network resources are represented by a finite set $\mathcal{R}$ of the feasible vectors describing the simultaneous achievable rates (in packets/slot) on the links. In general, it depends on inter-

[2]By definition, $A_{f,v}(\cdot) = 0$ when v is not in the set of source nodes of session f.

ferences among links. A resource allocation scheme then aims at choosing in each time-slot a rate schedule $\boldsymbol{S}(t) = (S_1(t), \ldots, S_L(t)) \in \mathcal{R}$, where $S_l(t)$ is the rate (in packet/slot) of link l. We assume that our time-slot length is chosen such that $S_l(t) = 0$ or 1. We model interference by a symmetric matrix $C \in \{0,1\}^{L \times L}$, where $C_{l,l'} = 1$ if link l' interferes with link l, and 0 otherwise. This generalizes the one-hop, two-hop, or generally K-hop interference model[3] popularly used in literature. We also assume that if $\boldsymbol{S} \in \mathcal{R}$ then $\boldsymbol{S}' \in \mathcal{R}$ if $\boldsymbol{S}' \leq \boldsymbol{S}$ coordinate-wise. The resource allocation scheme further shares the rate of each link among the various sessions. We denote by $S_{f,l}(t)$ the rate allocated for the session f over link l, with $\sum_{f \in \mathcal{F}} S_{f,l}(t) = S_l(t)$. Denote by $R_{\max}$ the maximum amount of packets per slot that one node may receive (resp. transmit) from (resp. to) its neighboring nodes. We will later use the constant $\Omega = A_{\max} + 2R_{\max}$.

We will consider problems in two types of systems. In *systems without congestion control*, there is no way to control the arrivals $A_{f,v}(t)$. All the arriving packets are queued and must be processed by the network. In *systems with congestion control*, one can control the arrival rates, which are typically outside the stability region.

Queueing dynamics. Denote by $Q_{f,v}(t)$ the queue length of session f at node v at time t. The queueing dynamics is defined by the following recursion:

$$Q_{f,v}(t+1) = Q_{f,v}(t) - \sum_{l \in O(v)} S_{f,l}(t) + \sum_{l \in I(v)} S_{f,l}(t) + A_{f,v}(t), \quad (1)$$

where $O(v)$ and $I(v)$ denote the sets of outgoing and incoming links of node v, respectively. Note that in the above recursion, we assume that the scheduling algorithm is aware of the queue lengths and does not schedule empty queues.

Stability. We use the following notion of queue stability:

DEFINITION 2.1 (STABILITY). *The system is stable, if*

$$\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t} \mathbb{E}\Big[\sum_{f \in \mathcal{F}, v \in \mathcal{V}} Q_{f,v}(\tau) \Big] \quad < \quad \infty.$$

When the queue length process $\boldsymbol{Q}(t)$ is Markovian, the system stability is equivalent to the positive-recurrence of $\boldsymbol{Q}(t)$ under the assumption of its aperiodicity and irreducibility.

2.2 Performance Metrics

(1a) Throughput. When arrivals are not controlled (i.e., without congestion control), a primary performance objective of any resource allocation scheme is to guarantee stability whenever possible, i.e., whenever the arrival vector $\boldsymbol{\lambda}$ belongs to the throughput region defined as follows:

DEFINITION 2.2 (THROUGHPUT-REGION). *The throughput-region $\Lambda \subset \mathbb{R}_+^{F \times V}$ is the set of all arrival rate vectors for which there exists a resource allocation scheme stabilizing the system.*

The throughput-region has already been characterized (see, e.g., [3, 15]) as follows: Denote by $\mathrm{co}(\mathcal{Y})$ the smallest convex set containing $\mathcal{Y}$. Then $\boldsymbol{\lambda} \in \Lambda$ if there exists a positive vector $\boldsymbol{s} = (s_{f,l}, f \in \mathcal{F}, l \in \mathcal{L})$ such that *(i)* for all f, if v is not in the set of destination nodes of session f, $\lambda_{f,v} + \sum_{l \in I(v)} s_{f,l} = \sum_{l \in O(v)} s_{f,l}$; *(ii)* there exists a vector $\boldsymbol{s}' = (s'_1, \ldots, s'_L) \in \mathcal{R}$, such that for all link l, $\sum_f s_{f,l} < s'_l$. Note that $s_{f,l}$ may be interpreted as the long-term rate allocated to session f along link l.

To compare various resource allocation schemes with respect to their achieved throughput-region, we use the notion of γ-throughput optimality:

DEFINITION 2.3 (γ-THROUGHPUT-OPTIMALITY). *A resource allocation scheme is γ-throughput-optimal for some $0 < \gamma \leq 1$, if it stabilizes the system for any $\boldsymbol{\lambda} \in \gamma\Lambda$.*[4]

(1b) Utility. With congestion control, a standard performance objective is to maximize an increasing and concave utility function that captures fairness, traffic elasticity, or user satisfaction, while maintaining system stability:

$$\begin{array}{ll} \text{Maximize} & \sum_{f,v} U_{f,v}(x_{f,v}) \\ \text{Subject to} & \boldsymbol{x} = (x_{f,v}, f \in \mathcal{F}, v \in \mathcal{V}) \in \Lambda, \end{array} \quad (2)$$

where $x_{f,v}$ is the long-term session rate, i.e., $x_{f,v} = \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} A_{f,v}(t)$. Note that we restrict our attention to stationary ergodic policies so that the above limit always exists.

(2) Delay. For systems with or without congestion control, we also consider the sum of the stationary queue lengths, i.e., $\sum_{f,v} \mathbb{E}[Q_{f,v}(t)]$.

2.3 Complexity Metrics

Different complexity metrics have been considered differently for centralized and distributed algorithms. In this paper, we focus on *time complexity*, which has particularly interesting interactions with concepts based on the "time" axis, such as stability and delay. In centralized algorithms, time complexity is the number of "steps" taken to solve the instance of a problem, where one step refers to one "basic" operation in a computer. In distributed algorithms, time complexity is measured by the number of "rounds", where one round corresponds to one unit of distributed operation, e.g., the time to communicate with neighboring links once.

We denote by χ (amortized) time complexity, i.e.,

$$\chi = \lim_{T \to \infty} \frac{\sum_{t=1}^{T} \chi(t)}{T},$$

where $\chi(t)$ is the instantaneous time complexity at time t. We assume that $\chi(t)$ is normalized by the data transmission time at each slot. In particular, for distributed algorithms, we assume that each time-slot is divided into control mini-slots and data transmission slot, in which case the number of mini-slots corresponds to time complexity $\chi(t)$ for slot t. A control message in distributed algorithms is assumed to include any information about queue lengths and link states available at a node until Section 5. In Section 6, we deal with the case when a control message is of a bounded size.

2.4 Throughput Optimal Resource Allocation

Throughput-optimal resource allocations have been proposed, e.g., in [17, 25] with slightly different technical assumptions and system models. Under these models, a resource allocation scheme consists of routing and scheduling. Routing chooses the sessions to be served on a link, and scheduling is responsible for selecting the schedule among the links. The resource allocation scheme, described in Algorithm 2.4, is well-known to be throughput-optimal.

[3] In this model, two links l, l' can transmit successfully only if they are at a distance greater than K hops.

[4] For simplicity, we just use the term 'throughput-optimal' when $\gamma = 1$.

Algorithm 1 TORA (Throughput Optimal Resource Allocation)

- ***Routing (Max-Differential-Backlog).*** Let $\mathrm{tx}(l)$ (resp. $\mathrm{rx}(l)$) be the source (resp. destination) of the link l and also let $D_{f,l} = Q_{f,\mathrm{tx}(l)}(t) - Q_{f,\mathrm{rx}(l)}(t)$. On the link l, the session $f_l^\star$ chosen for routing is:

$$f_l^\star = \arg\max_{f\in\mathcal{F}} D_{f,l}. \quad (3)$$

  Let $Q_l(t) = \max_{f\in\mathcal{F}} D_{f,l}$, which is henceforth referred to as just "queue-length" over the link l.
- ***Link scheduling (Max-Weight).*** Select the rate schedule $\boldsymbol{S}^\star(t)$ defined by:

$$\boldsymbol{S}^\star(t) \quad = \quad \arg\max_{\boldsymbol{S}\in\mathcal{R}} \sum_{l\in\mathcal{L}} Q_l(t)S_l. \quad (4)$$

We denote by $W^\star(t)$ the maximum weight over all possible schedules (i.e., the weight of $\boldsymbol{S}^\star$):

$$W^\star(t) \quad = \quad \max_{\boldsymbol{S}\in\mathcal{R}} \sum_{l\in\mathcal{L}} Q_l(t)S_l.$$

In what follows, for an arbitrary resource allocation scheme, we use $W(t)$ to denote the weight of the rate schedule $\boldsymbol{S}(t)$ at time t, i.e., $W(t) = \sum_{l\in\mathcal{L}} Q_l(t)S_l(t)$.

In TORA, routing can be performed using only local information with very small complexity. However, the MW algorithm for link scheduling require high complexity, due to the fact that it can be reduced to WMIS (Weighted Maximum Independent Set) problem, which is NP-hard[5]. This high complexity should also become an impediment to distributed implementation. Thus, we mainly focus on link scheduling in the rest of the paper.

3. GENERALIZED FRAMEWORK

3.1 (γ, ξ, χ)-Approximate Algorithms

To study complexity-stability-delay trade-offs, we first classify algorithms according to *(i)* how far they are from making the "correct" scheduling decisions, and *(ii)* their complexity.

DEFINITION 3.1 ((γ,ξ)-APPROXIMATE ALGORITHMS). *A link scheduling is said to be (γ,ξ)-approximate for some finite, positive constants γ and ξ, if, at any time-slot t,*

$$W(t) \geq G(t)W^\star(t) - C(t),$$

where $G(t)$ and $C(t)$ are random variables, such that

$$G(t) \geq \gamma, \quad \mathbb{E}[C(t) \mid \boldsymbol{Q}(t)] \leq \xi.$$

We henceforth say that an (γ,ξ)-approximate algorithm with complexity less than or equal to χ is (γ,ξ,χ)-approximate. However, we just use (γ,ξ), unless χ is explicitly needed. Note that our definition of (γ,ξ)-approximate algorithms is general in a sense that $C(t)$ are random variables. This allows us to handle wide classes of the existing link scheduling algorithms such as randomized ones.

[5]For one-hop interference model, it is not NP-hard, and the problem can be solved with $O(L^3)$ complexity. However, unless explicitly stated, we henceforth assume that optimal link scheduling requires exponential complexity.

3.2 Examples

We now provide examples of approximate algorithms. Proofs of Lemmas in this section are delayed in the Appendix.

We first introduce some necessary notations. Let $\Theta(l)$ be the set of links interfering with the link l, and $\theta(l)$ be the maximum number of links in $\Theta(l)$ that can be scheduled simultaneously without conflict. Finally, we let $\theta = \max_{l\in\mathcal{L}} \theta(l)$. Note that $\Theta(l)$, $\theta(l)$, and θ depend on the given graph G as well as the interference model, but we omit it from the notations for simplicity.

(a) Greedy scheduling [12,23,27]. In the Greedy scheduling, 1) start with an empty schedule and a set $\mathcal{N} = \mathcal{L}$; 2) add to the schedule the link $l \in \mathcal{N}$ with the *longest* queue length, and remove from $\mathcal{N}$ the links interfering with link l; 3) repeat step 2) until $\mathcal{N}$ is empty. The Greedy scheduling is $(1/\theta, 0)$-approximate with the MW scheduling [23].

(b) Locally-Greedy scheduling. The Locally-Greedy scheduling is similar to the Greedy scheduling, except that in each step 2), the link l to be added to the schedule is chosen randomly among the links in $\mathcal{N}$ that have a *locally-longest* queue length in $\mathcal{N}$. A link is said to have a locally-longest queue length in $\mathcal{N}$ if its queue length is greater than any of its interfering links also in $\mathcal{N}$.

LEMMA 3.1. *The Locally-Greedy scheduling is $(1/\theta, 0)$-approximate.*

The best complexities of Greedy and Locally-Greedy algorithms known to date are $O(L\log V)$ and $O(L)$, respectively, for one-hop interference [18]. This asymptotic decrease in complexity is intuitive, since Greedy $\subset$ Locally-Greedy.

(d) Pick-and-Compare-γ [4,7,14,20,24]. The algorithm first generates a random schedule $\boldsymbol{S}'(t)$ satisfying **C1**, and then schedule $\boldsymbol{S}(t)$ defined in **C2**.

C1. $\exists 0 < \delta \leq 1$, s.t. $\mathbb{P}[\boldsymbol{S}'(t) = \boldsymbol{S}|\boldsymbol{Q}(t)] \geq \delta$, for some schedule $\boldsymbol{S}$, where $W(\boldsymbol{S}) \geq \gamma W^\star(t)$.

C2. $\boldsymbol{S}(t) = \arg\max_{\boldsymbol{S}=\{\boldsymbol{S}(t-1),\boldsymbol{S}'(t)\}} W(\boldsymbol{S})$.

The Pick-and-Compare-γ provides only a probabilistic guarantee of finding a "γ-optimal" schedule in **C1** with selection of "better" schedule in **C2**. The idea of "pick" and "compare" has been used in many sophisticated distributed scheduling algorithms that achieve throughput-optimality with polynomial complexity. First, the following lemma states that the Pick-and-Compare scheduling belongs to the framework of (γ,ξ)-approximate algorithms.

LEMMA 3.2. *A Pick-and-Compare-γ scheduling is $(\gamma, \Omega V(1+\gamma)/\delta)$-approximate.*

This result has been proved for $\gamma = 1$ in [2]. The proof for $\gamma < 1$ is presented in the Appendix.

3.3 Throughput and Delay Performance

The three parameters γ, ξ, and χ can be controlled by system designers. As it will be shown in Theorem 3.1, the stability property depends on γ only. We will further show that ξ closely relates to delay, whereas by definition χ is directly related to time complexity. We can choose two parameters (thus, two performance metrics) freely, but the remaining performance metric will be affected by such a choice of two parameters. Later in Section 4, we formally study the trade-offs among these three parameters. First, some basic characterizations on throughput and delay are as follows.

THEOREM 3.1 (THROUGHPUT). *(γ, ξ)-approximate algorithms are γ-throughput-optimal.*

The proof when $\gamma = 1$, $\xi = 0$ was presented in [17, 25]. Its extension to $\gamma < 1$, $\xi > 0$ is straightforward, and will be presented in the Appendix for completeness.

Now, we discuss the delay property. Consider a (γ, ξ)-approximate algorithm Π and an arrival vector $\boldsymbol{\lambda}$ stabilizable by Π, i.e., $\boldsymbol{\lambda} \in \gamma\Lambda$. We first introduce a notion of the distance between $\boldsymbol{\lambda}$ and the boundary of $\gamma\Lambda$, which essentially represents how heavily the system is loaded.

DEFINITION 3.2 (DISTANCE).

$$d_\gamma(\boldsymbol{\lambda}) = \sup\{\delta : \boldsymbol{\lambda} \in (1-\delta)\gamma\Lambda\}. \tag{5}$$

Analyzing the exact delay performance of the system considered in this paper is very difficult and still an open problem due to very complex coupling of queueing dynamics across links with large dimensions. Thus, we rely on the upper bound on the mean queue length (thus mean delay by Little's formula), as proved in Theorem 3.2. The delay bound for the MW scheduling (i.e., $\gamma = 1$) has already been studied in [7, 10, 15–17, 22] for slightly different system models, based on the ideas of telescoping the Lyapunov function values over times [13]. Again, extension to $\gamma < 1$ is not difficult, which is shown in the Appendix for completeness.

THEOREM 3.2 (DELAY). *Let Π be a (γ, ξ)-approximate algorithm, and $\boldsymbol{\lambda} \in \gamma\Lambda$. We have:*

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\sum_{f,v}\mathbb{E}\Big[Q_{f,v}(t)\Big] \quad \leq \quad \frac{f(\xi)}{\gamma d_\gamma(\boldsymbol{\lambda})}, \tag{6}$$

where $f(\xi) = VF\xi + (VF\Omega^2 V/2)$, i.e., the mean delay bound is linear *in ξ.*

4. COMPLEXITY-STABILITY-DELAY TRADE-OFF

4.1 Two Methods of Complexity Reduction

We first summarize two ways of reducing complexity and then quantify their impact on throughput and delay performance inside the framework of (γ, ξ, χ)-approximate algorithms.

4.1.1 Weight Approximation

As discussed earlier, optimal MW scheduling has to solve WMIS problems over time-slots with different weights. One natural way of complexity reduction is to take approximation algorithms to have polynomial time algorithm at the cost of sub-optimality in the resulting weight. The sub-optimality ratio is $\gamma < 1$ in our definition. Greedy and Locally-Greedy scheduling are constant-factor approximation algorithms, where all with approximation ratio is $1/\theta$. Then, from Theorem 3.1, its (worst-case) throughput region is reduced from Λ to Λ/θ. In regard to delay performance, consider a stabilizable arrival vector, i.e., $\boldsymbol{\lambda} \in \Lambda/\theta$. Weight approximation does not affect ξ, but delay will be affected by the change of throughput-ratio, i.e., $\gamma = 1 \to \gamma = 1/\theta$ and $d_\gamma(\boldsymbol{\lambda})$ from Theorem 3.2. We note that it is still an open problem to achieve *arbitrary* $\gamma < 1$ with just a polynomial time approximation algorithm, since the WMIS problem does not allow PTAS (Polynomial-Time-Approximation-Scheme) [26].

4.1.2 Frequency Reduction

Stretching. Another way of complexity reduction *without affecting the throughput region* is to compute optimal schedules infrequently. This idea of "stretching" has been presented in several recent works [2, 7, 22] and can be formally and more generally defined as follows:

DEFINITION 4.1 (m-STRETCHING). *Consider a resource allocation algorithm Π that is (γ, ξ, χ)-approximate, and select a sequence of random time slots $t_0, t_1, t_2, \ldots$, such that $\mathbb{E}[t_i - t_{i-1}] = m$, for all $i = 1, 2, \ldots$, and some fixed $m < \infty$. Now define the i-th frame as a period of $t_i - t_{i-1}$ consecutive time-slots. Then the m-stretched algorithm $\Pi(m)$ obtained from Π uses the same procedure as Π, but updates the optimal schedules only at the beginning slot of each frame. Once the schedule has been defined, it remains the same for the entire duration of the frame.*[6]

The amortized complexity of $\Pi(m)$ naturally becomes χ/m. Note that this stretching is well-defined but differently for both centralized and distributed algorithms, where χ is the number of "steps" and "rounds", respectively. Throughput and delay property of m-stretched algorithms are stated in the following theorem:

THEOREM 4.1. *For a (γ, ξ, χ)-approximate algorithm Π, $\Pi(m)$ is $(\gamma, \xi + mV\Omega(1+\gamma), \chi/m)$-approximate.*

It is well-known as well as intuitive from control theory and other related work [2, 7, 22][7] that stretching does not affect stability, which is also verified in view of our Theorems 3.1 and 4.1. This is due to the fact that stability is guaranteed as long as the *long-term* service rates are equal to the *long-term* arrival rates. Thus, infrequent computation of some scheduling with a finite period on average does not affect the stability property. However, it may adversely affect delay, which is the price to pay for complexity reduction. From Theorem 3.2 and Theorem 4.1, the delay increase by m-stretching is no larger than *linear* with m.

The following example shows that this linear increase with respect to m is tight in the worst-case in the sense that there exists a network topology where the delay of m-stretching increases linearly with m, as shown in Example 4.1. Note that this does not imply that the delay bound in Theorem 3.2 is tight. In Example 4.1, we start from a $(1, 0)$-approximate algorithm, i.e., MW scheduling, and stretch it by a factor m, i.e., $(1, O(m))$-approximate algorithms.

EXAMPLE 4.1. *Consider a network of two interfering links (thus, only one link should be activated for successful transmission) and two single-hop sessions over these links. For both sessions, packets are generated according to i.i.d. Bernoulli processes of identical rates λ. The service rate is one packet per slot. Then, the throughput region is characterized by $\lambda < 1/2$. Assume now that $\lambda < 1/2$. When m is large, by ergodicity, the m-stretched algorithm serves the two queues alternatively for m consecutive slots, with high probability: Queue 1 is served in slots $2km, \ldots, (2k+1)m-1$, and Queue*

[6] We can define this stretching in a slightly different manner that computation of an optimal schedule at t_i is equally divided over the entire i-frame. The amortized complexity is same for both definitions.

[7] There, stretching is deterministic, i.e., $t_i = mi$, $i = 1, 2, \ldots$.

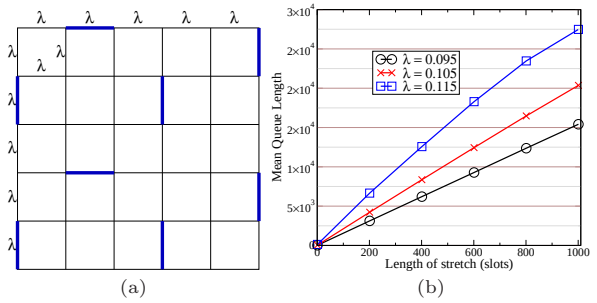

Figure 1: (a) A 5x5 grid network and an example of maximal schedule under 2-hop interference model. (b) Mean queue length vs. stretching parameter m.

2 is served in slots $(2k+1)m, \ldots, (2k+2)m-1$. *In this case, a lower bound on the mean delay of packets at Queue 1 is obtained assuming that all packets arriving at this queue between slots* $2(k-1)m$ *and* $2km$ *are actually generated in slot* $2km$. *It is then easy to see that the mean delay in this modified system is* $O(\frac{m}{(1/2-\lambda)})$.

Now, we provide another numerical example with more complex network topology. We again start from the MW scheduling and stretch it by increasing m. Consider now a 5×5 grid network with single-hop sessions and 2-hop interference model. In this grid, a line between a pair of lattice points corresponds to a link. Figure 1(a) shows the topology and an example maximal schedule in the system. Each link supports a session, and for all sessions, packets are generated according to Bernoulli processes at the same rate λ packets per slot. This network has 40 links (i.e., sessions) and 1923 maximal schedules. Numerically, we observe that the maximum throughput is achieved for $\lambda \approx 0.12$. In Figure 1(b), we present simulation results showing the mean total queue length as a function of the stretching parameter m, for $\lambda = 0.095, 0.105$ and 0.115. Again the queue length linearly increases with m.

Stretching and Pick-and-Compare scheduling. The idea of (randomized) stretching is important to understand complexity reduction of the Pick-and-Compare scheduling. From Lemma 3.2 and Theorem 4.1 ($\xi = 0$), m-stretching and the Pick-and-Compare with δ in **C2** are equivalent, where $m = 1/\delta$. In fact, the Pick-and-Compare scheduling is a version of $1/\delta$-stretching, where a sequence of slots (t_i) are determined by a geometric random variable with parameter δ. We comment that the comparison operation **C2** is crucial to develop a distributed algorithm based on the Pick-and-Compare scheduling. This is because it is very difficult to know in a distributed manner that a random schedule at a time-slot is indeed an optimal schedule. The operation **C2** provides stronger guarantee than the condition of stretching that the same schedules are used inside a frame, but is more amenable to distributed implementation.

4.2 3D Trade-offs: Three Pairwise Trade-offs

4.2.1 Complexity-Delay Trade-off

Theorems 3.2 and 4.1 allow us to quantify the complexity-delay trade-off realized by infrequent γ-optimal schedule computation, when *the stability region (i.e., γ) is* fixed.

We exemplify this trade-off in the case of the MW scheduling and the Pick-and-Compare-1 with the parameter δ (see

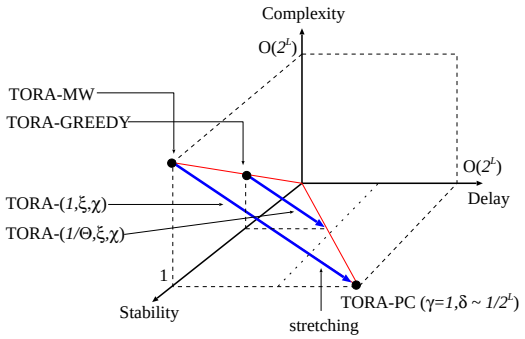

Figure 2: 3D trade-off space, and the achieved trade-off points and curves by some of the algorithms with the MW rule. PC: Pick-and-Compare-1.

condition **C1** in Section 3), where recall that both algorithms are throughput optimal. Then to reduce exponential complexity of the MW scheduling to a polynomial complexity, one has to stretch the algorithm by a factor $\sim 2^L$, in which case the average delay scales as 2^L. The Pick-and-Compare scheduling is known to have a polynomial complexity only if its parameter δ is close to 2^{-L}, in which case the average delay also scales as 2^L (in view of Lemma 3.2 and Theorem 3.2). Exponential decay of probability of finding the optimal schedule is supported by the existing distributed algorithms [4, 7, 14, 20, 24] based on the Pick-and-Compare scheduling. The price to pay to reduce complexity (from exponential to polynomial) with the above algorithms is large. This is due to the fact that we need an exponential stretching, and thus, an exponential growth of delay.

4.2.2 Stability-Delay Trade-off

Now we *fix the complexity,* and analyze the trade-off between stability and delay. This analysis aims at answering the following question: for a fixed complexity, *could we have incentives in terms of delay to choose a γ-throughput optimal algorithm with $\gamma < 1$ rather than throughput optimal algorithms?.*

Assume that the arrival vector $\boldsymbol{\lambda} \in \Lambda$, and denote by $\underline{\gamma}$ the smallest $\gamma > 0$ such that $\boldsymbol{\lambda} \in \gamma\Lambda$. Then, to stabilize the system, we may choose any γ-throughput optimal algorithm for $\gamma > \underline{\gamma}$.

Consider two algorithms Π and Π' that are $(1, 0, \chi)$- and $(\gamma, 0, \chi')$-approximate, respectively, with $\underline{\gamma} < \gamma < 1$. Of course, we should expect that $\chi' < \chi$. Now we *equalize the complexities* of Π and Π' by stretching these algorithms by some factors. Here, without loss of generality, we stretch Π and Π' by factors χ and χ', so as to have unit complexity. Then the delay bounds for these stretched algorithms $\Pi(\chi)$ and $\Pi'(\chi')$ are respectively:

$$\frac{2\chi\Omega V + \Omega^2 V/2}{d_1(\boldsymbol{\lambda})}, \quad \frac{2\chi'\Omega V + \Omega^2 V/2}{\gamma d_\gamma(\boldsymbol{\lambda})}.$$

Since $\gamma d_\gamma(\boldsymbol{\lambda}) = \gamma - 1 + d_1(\boldsymbol{\lambda})$ by Definition 3.2, we conclude that $\Pi'(\chi')$ provides a better delay performance than that of $\Pi(\chi)$ if:

$$d_1(\boldsymbol{\lambda})\chi' + \frac{(1-\gamma)\Omega}{4} < (d_1(\boldsymbol{\lambda}) - (1-\gamma))\chi. \qquad (7)$$

We illustrate the trade-off by comparing the MW scheduling Π, which is $(1, 0, \chi \sim 2^L)$-approximate, and the Greedy

scheduling Π', which is $(1/\theta, 0, \chi' \sim \text{polynomial})$-approximate. We can easily see that we get better delays with the Greedy scheduling (i.e., (7) holds) when the network size L, is sufficiently large.

4.2.3 Complexity-Stability Trade-off

Now, we *equalize the delays* (again by stretching), and then study the relationship between complexity and stability for two scheduling algorithms. This allows us to know whether choosing an algorithm with smaller stability region can be beneficial in terms of complexity.

Consider the same arrival rate $\boldsymbol{\lambda}$, and the same algorithms Π and Π' as those considered for the analysis of the stability-delay trade-off. Now, stretch Π by factor m, such that the delays of $\Pi(m)$ and Π' are equalized. Then, m should be chosen as

$$m = \frac{(1-\gamma)\Omega}{4(d_1(\boldsymbol{\lambda}) - (1-\gamma))}.$$

The complexities of $\Pi(m)$ and Π' become, respectively,

$$\chi \frac{4(d_1(\boldsymbol{\lambda}) - (1-\gamma))}{(1-\gamma)\Omega}, \quad \text{and} \quad \chi'.$$

Therefore, Π' is simpler, in terms of time complexity, than $\Pi(m)$ if $\chi' < \chi \frac{4(d_1(\boldsymbol{\lambda})-(1-\gamma))}{(1-\gamma)\Omega}$. Again, if we compare the stretched MW scheduling $\Pi(m)$ and the Greedy scheduling Π', we see that Π' provides better complexity than $\Pi(m)$ while at the same delays.

To summarize, we may indeed decide to choose a γ-throughput optimal algorithm with $\gamma < 1$, as a way to reduce delays (for a fixed complexity), or as a way to reduce complexity (for a fixed delay), as long as the network size is reasonably large.

4.3 Understanding Alternative Approaches

Now we turn to understanding the alternative approaches of getting arbitrarily close to throughput-optimality with polynomial time complexity.

Mixture of weight approximation and stretching. Examples include [20] under one-hop and its extension to general K-hop interference model [29]. The algorithms, parameterized by k, are $(k/(k+2), O(k^L), 4k+2)$-approximate[8], thereby delay scales as k^L. Getting throughput arbitrarily close to 1 with polynomial time complexity cannot be achieved just by taking weight-approximating algorithms. This is due to the fact that WMIS does not allow PTAS (Polynomial-Time-Approximation-Scheme), as mentioned in Section 4.1.1, where, however, the work in [20,29] essentially mix weight-approximation with stretching (realized by the Pick-and-Compare scheduling) to achieve it.

Weight approximation for restricted topology. The authors in [7, 19] proposed the family of algorithms, also parameterized by k, with all polynomial complexity as well as polynomial delay, w.r.t., k. This may seem to contradict our analysis, but they considered special cases of network topologies, where the WMIS problem allows PTAS. Thus, without stretching, they can parameterize the algorithms whose stability is arbitrarily close to that of the optimal one, even with polynomial delay. As an example, in [7], non-expandable network topology, which we refer the readers to [7] for its formal definition, is considered. The work in [8, 21] has the similar idea, where the optimal scheduling is implemented with polynomial time complexity [8] or the performance of maximal scheduling is improved [21], both under tree-based topologies.

[8]In [20], $O(k^L)$ is the order of the *lower-bound* of $1/\delta$ of the Pick-and-Compare scheduling. In this discussion, we assume such a lower-bound is order-wise tight.

5. COMPLEXITY-UTILITY-DELAY TRADE-OFF

When the arrival rates are outside of the throughput-region, congestion control algorithm needs to be used in conjunction with routing and link scheduling, often in the utility maximization framework. This section is devoted to the trade-off study among utility, complexity, and delay.

5.1 Utility-Optimal Resource Allocations

First, we briefly review an optimal algorithm that maximizes the achieved utility and also stabilizes the system. Various types of algorithms such as dual-based algorithms and primal-dual-based algorithms, have been proposed under slightly different conditions and system models (see [28] and the references therein for the survey). In this paper, we use the dual-based algorithm in [15], shown in Algorithm 2, to study the tradeoff.

Algorithm 2 UORA (Utility Optimal Resource Allocation)

- ***Congestion control.*** Each source node v of each session f set their data injection rate $A_{f,v}$ to be the optimal solution of the following:

$$\max_{A_{f,v}(t) \le A_{max}} \Big(\beta U_{f,v}(A_{f,v}(t)) - A_{f,v}(t) Q_{f,v}(t)\Big), \quad (8)$$

  where $\beta > 0$ is a parameter.
- ***Routing and Link Scheduling.*** Same as those in TORA.

The β corresponds to the tunable parameter which determines the difference value $U^\star - \bar{U}(\beta)$, where $U^\star$ the optimal utility (i.e., the optimal solution of (2)). The $\bar{U}(\beta)$ is the achieved utility by UORA-MW for the parameter β, i.e., $\bar{U}(\beta) = U(\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} A_{f,v}(t))$, where $A_{f,v}(t)$ is controlled by (8). In [15], the authors proved that as β increases, $\bar{U}(\beta)$ approaches to $U^\star$, but the delay may linearly increase.

5.2 3D Trade-off

Consider a UORA algorithm, where link schedule is replaced by a (γ, ξ)-approximate algorithm, which we denote by UORA(γ, ξ). This enables us to quantify the impact of complexity on utility and delay. Denote by $U^\star(\gamma)$ the optimal solution of (2), where the constraint set is changed from Λ to the throughput-region attained by a (γ, ξ)-approximate algorithm, i.e., $\gamma\Lambda$. Unfortunately, it is known that UORA(γ, ξ) does not converge to $U^\star(\gamma)$ unless $U_{f,v}(\cdot)$ is linear. In other words, we require the property of the utility function that $U'(\gamma x) = \gamma U'(x)$, for $x > 0$. We refer the readers to an example of such a case to [12].

Therefore, in this section we focus only on $(1, \xi)$-approximate algorithms, such as the MW scheduling or the Pick-and-Compare scheduling-1. We extend the notation $\bar{U}(\beta)$ to $\bar{U}(\beta, \xi)$ to refer to the achieved utility by UORA$(1, \xi)$ with parameter β. Denote $\bar{Q}$ be the average queue length, i.e., $\bar{Q} \triangleq \limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\sum_{f,v} Q_{f,v}]$.

THEOREM 5.1. *UORA$(1,\xi)$ stabilizes the system for any $\xi < \infty$, and*

$$\bar{Q} \leq \frac{B+\xi+\beta V U_{max}}{\lambda_{max}}, \quad \bar{U}(\beta,\xi) \geq U^\star - \frac{B+\xi}{2\beta},$$

where B is some constant depending on the network size, $\lambda_{max} = \sup_\lambda\{(\lambda)_{f,v} \in \Lambda\}$, and U_{max} is an upper bound on the utility, i.e., $U_{max} = \max_{v\in\mathcal{V}, x_{f,v}\leq A_{max}} \sum_f U_{f,v}(x_{f,v})$.

The proof is presented in the Appendix. Theorem 5.1 says that the average delay scales linearly with the parameter ξ, as in the system without congestion control. *However, the achieved utility is also* affected by ξ *linearly,* which is in sharp contrast to the stabilizable inputs, where ξ does not affect throughput (see Theorem 3.1). From Theorem 5.1, we can quantify various 3D trade-offs as in the case without congestion control. Due to space limitation, we only provide one example here.

Consider UORA and the Pick-and-Compare-1 with the parameter δ. To have a polynomial-time algorithm, from Lemma 3.2, we require that $\delta = 1/O(2^L)$, thereby $\xi \sim 2^L$ in Theorem 5.1. This implies that for any choice of β, delay scales exponentially with the network size, and (ii) by choosing $\beta \sim 2^L$ large enough, we can have polynomial sub-optimality (w.r.t. $U^\star$) of the achieved utility, but at the expense of much-increased delay $\sim O(\xi+\beta) = 2^L+2^L = 2^{L+1}$.

6. EFFECTIVE THROUGHPUT UNDER BOUNDED MESSAGE SIZE

6.1 Impact of Bounded Message Size

In this section, we consider the important case of *distributed implementation* of those (γ,ξ,χ)-approximate algorithms that use the *queue-lengths* to determine a schedule. The example algorithms addressed earlier belong to this category. They exchange control messages to notify the neighbors of their queue lengths. Recall that $\chi(t)$ is the time for control message exchange at time t, measured by the number of unit distributed operation such as convey the queue length over a link to its neighbor. We remove somewhat impractical assumption in earlier sections that a control message can include arbitrary size of state information. Indeed, a control message size is usually bounded due to scarce resource of communication links and implementation issues. The overhead of control messages should not be ignored in a proper accounting of the tradeoff study.

To study practical impact of bounded message size on scheduling, we introduce the notion of *effective throughput.* Effective throughput corresponds to the real throughput of data that considers time resource occupied by control message exchange. A little thought leads us to the fact that the effective throughput region of a (γ,ξ,χ)-approximate algorithm decreases from $\gamma\Lambda$ to $(\frac{\gamma}{1+\chi})\Lambda$.

A simple implementation of queue-size based distributed scheduling algorithms may use a "bit-encoder" which encodes $Q_l(t)$, for a link l, with $\log_2(Q_l(t)+1)$ bits. When the message is one-bit, this will generate $\log_2(Q_l(t)+1)$ number of control messages. Observe that $\chi(t)$ is an increasing function of $\boldsymbol{Q}(t)$, and tends to ∞, as $Q_l(t) \to \infty$. Then, this system using the bit-encoder becomes unstable for any positive arrival rate, since as the queue length increases, the service rate (per unit time) goes to zero. Thus, the effective throughput of such systems becomes *zero*, irrespective of network size and (positive) arrival rate.

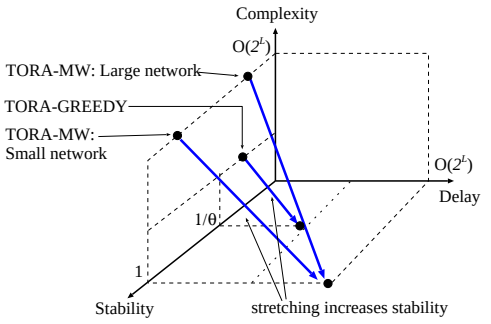

Figure 3: The achieved trade-off points for MW and Greedy with bounded message size and effective throughput.

6.2 Time-Differential Queue Size and Resulting 3D-Tradeoffs

The root cause of bad effective throughput for bounded message size is due to the fact that time complexity depends on queue-lengths, which can be arbitrarily large. To achieve a positive throughput-region, we propose the idea of using *time-differential queue size*, i.e., *only the increase or decrease* in queue size (with initialization of updating the queue lengths at time slot 0), whenever the actual queue size information is needed in the "original" algorithms. Exchanging the queue length difference over time allows the algorithms to have time-complexity that does not increase with the queue lengths. The following theorem states that complexity in the algorithms with time-differential queue size does not change significantly, compared to those which uses actual queue lengths:

THEOREM 6.1. *For a original $(\gamma,0,\chi)$-approximate algorithm Π, its m-stretched algorithm Π' with time-differential queue size is $(\gamma, mV\Omega(1+\gamma), \chi\log_2(m\Omega V)/m)$-approximate.*

The proof is not hard, since χ (which is independent of boundedness of message size) is reduced by a factor of m because the γ-optimal schedule is computed over m slots on average, and the amount of queue-size change over a link is bounded by $m\Omega V$. For example, for the original MW scheduling, which is $(1,0,\chi)$-approximate, the MW scheduling and the Pick-and-Compare-1 with parameter δ using time-differential queue size is $(1,0,\chi\log_2(\Omega V))$, and $(1, 2V\Omega/\delta, \chi\delta\log_2(\Omega V/\delta))$-approximate.

Now, we discuss the tradeoffs in throughput, delay, and complexity due to introduction of effective throughput with bounded message size. The achievable 3D tradeoff is not same as before, as illustrated in Figure 3, and summarized as follows:

(i) Stretching affects the effective throughput. From Theorem 6.1, the effective throughput region of Π' is $\gamma/(1+\chi\log_2(m\Omega V)/m)\Lambda$. Effective throughput region can become arbitrarily close to its ideal throughput, i.e., γ, as $m \to \infty$, which, again, leads to an exponential increase of delay.

(ii) Effective throughput of low cost scheduling via weight approximation may be larger or smaller than that of its optimal algorithm, depending on the network size. To illustrate it, consider the MW rule and Greedy schedul-

ing. From Theorem 6.1, their effective throughputs are $1/\chi^{MW}\log_2(\Omega V)$ and $\gamma/\chi^{GR}\log_2(\Omega V)$, respectively, where $\chi^{MW} = O(2^L)$ and $\chi^{GR} = O(L\log V)$, and $\gamma = 1/\theta < 1$. Thus, if $1/\chi^{MW} > 1/\theta\chi^{GR}$, the effective throughput in the MW rule is larger, which holds for sufficiently small-scale networks, and the opposite result is obtained for sufficiently-large scale networks.

7. CONCLUDING REMARKS

Scheduling in wireless networks can be conducted in many variants. Comparing these alternatives requires understanding not just throughput region but also the trade-off among various performance metrics with complexity. This paper develops a generalized framework, and characterizes the achievable trade-off curves in a parameterized family of algorithms that cover major classes of known algorithms. Pairwise trade-offs are proved, and extensions to complexity-utility-delay trade-off are developed. These characterizations quantify intuitions on what price must be paid for simplicity of scheduling algorithms.

Our study also reveals many under-explored questions in the important area of wireless scheduling: How to use *two* degrees of freedom in parameterization of scheduling algorithms so that an entire *surface* rather than just a *curve* as in current works can be traced out in the 3D trade-off space? How to incorporate other measures of complexity such as communication complexity? Can "outer bounds" on trade-off surface be obtained from "converse theorems"? Finally, how to model and characterize the impacts of practical issues, such as bounded message size, so that distributed scheduling can be actually used in practice?

Appendix

Proof of Lemma 3.1. Choose any optimal schedule $S^\star$ with MW rule. Let $(S_1, S_2, \ldots, S_m)$ be the sequence of schedules when a link is sequentially added by Locally-greedy scheduling. We also let $S_i^\star = S^\star \cap (S_i \cup_{l\in S_i} \Theta(l))$, where recall that $\Theta(l)$ is the set of interfering links with l. It suffices to show that $W(S_i) \geq \frac{1}{\theta}W(S_i^\star)$, $i = 1, \ldots, m$, since the final schedule S_m is a maximal schedule and satisfies $W(S_m^\star) = W(S^\star)$. Let l_i be the added link at i-th phase, i.e., $l_i = S_i \setminus S_{i-1}$. Remarking that the number of links in $S^\star \cap \Theta(l_i)$ is no larger than θ and the weight of l_i is the largest of the links in $\Theta(l_i)$ (i.e., locally-longest), the sum of weights of links in $S^\star \cap \Theta(l_i)$ is no larger than $\theta W(\{l_i\})$, the result follows. □

Proof of Lemma 3.2. Denote by $\{t_i\}_{i=0,1,\ldots}$ be the sequence of (random) time slots, where $W(t_i) \geq \gamma W^\star(t_i)$. Let $\Delta t_i = t_{i+1} - t_i$. Now for $\forall t \in (t_i, t_{i+1})$, we have: (i) $W(t) \geq W(t_i) - V\Omega$ and (ii) $W^\star(t) \leq W^\star(t_i) + \Delta t_i V\Omega$. Hence, $W(t) \geq \gamma W^\star(t) - \Delta t_i V\Omega(1+\gamma)$. Thus, the result follows, from the fact that $\mathbb{E}[\Delta t_i] = 1/\delta$. □

Proof of Theorem 3.1 and Theorem 3.2. We first introduce the following notations: All quantities associated with session f are denoted by a superscript f: $\boldsymbol{Q}^f(t) = (Q_{f,v}(t), v \in \mathcal{V})$, $\boldsymbol{S}^f(t) = (S_{f,l}(t), l \in \mathcal{L})$, $\boldsymbol{A}^f(t) = (A_{f,v}(t), v \in \mathcal{V})$, and $\boldsymbol{\lambda}^f = (\lambda_{f,v}, v \in \mathcal{V})$. For a vector $\boldsymbol{s} = (s_{f,l}, f \in \mathcal{F}, l \in \mathcal{L})$, we define $s_l = \sum_{f\in\mathcal{F}} s_{f,l}$. We also define the $V \times L$ matrix R^f with (v,l)-component equal to 1 if $v = rx(l)$ and v is not the destination of packets of session f, -1 if $v = tx(l)$, and 0 otherwise. With these notations, the evolution of queues related to session f can be written as:

$$\boldsymbol{Q}^f(t+1) \;=\; \boldsymbol{Q}^f(t) + R^f\boldsymbol{S}^f(t) + \boldsymbol{A}^f(t).$$

Also note that the weight of the schedule $\boldsymbol{S}(t)$ becomes:

$$W(t) \;=\; -\sum_{f\in\mathcal{F}} \boldsymbol{Q}^f(t)^T R^f \boldsymbol{S}^f(t),$$

where $\boldsymbol{a}^T$ is the transpose of vector $\boldsymbol{a}$. One can easily show (as in [25]) that if $\lambda \in \gamma\Lambda$, then there exists $\delta > 0$ such that

$$\forall i \in \mathcal{I}, \exists \boldsymbol{s}(i) = (s_{f,l}(i), f \in \mathcal{F}, l \in \mathcal{L}) : (s_1(i), \ldots s_L(i)) \in \mathcal{R}_i,$$
$$\text{and } \boldsymbol{\lambda}^f = -(1-\delta)\gamma R^f \sum_{i\in\mathcal{I}} \pi_i \boldsymbol{s}^f(i), \forall f \in \mathcal{F}. \quad (9)$$

Recall that the distance between $\boldsymbol{\lambda}$ and $\gamma\Lambda$ has been defined by the supremum of δ such that (9) holds. Now consider the usual quadratic Lyapunov function and the corresponding drift: $L(t) = \sum_{f,v} Q_{f,v}(t)^2$ and $\Delta L(t) = E[L(t+1) - L(t)|\boldsymbol{Q}(t)]$. Following [25], we can show that:

$$\Delta L(t) \leq \sum_f E[(R^f\boldsymbol{S}^f(t) + \boldsymbol{A}^f(t))^T (R^f\boldsymbol{S}^f(t) + \boldsymbol{A}^f(t))|\boldsymbol{Q}(t)]$$
$$+ 2\boldsymbol{Q}^f(t) E[(R^f\boldsymbol{S}^f(t) + \boldsymbol{A}^f(t)^T|\boldsymbol{Q}(t)]. \quad (10)$$

The first term at the right side of (10) can be easily bounded by a constant $b = V\Omega^2$. The second term is:

$$2\sum_f \boldsymbol{Q}^f(t)^T R^f \mathbb{E}[\boldsymbol{S}^f(t)|\boldsymbol{Q}(t)] + 2\sum_f \boldsymbol{Q}^f(t)^T \boldsymbol{\lambda}^f.$$

Now for (γ, ξ)-approximate algorithms:

$$\sum_f \boldsymbol{Q}^f(t)^T R^f \mathbb{E}[\boldsymbol{S}^f(t)|\boldsymbol{Q}(t)] \leq -\gamma\mathbb{E}[W^\star(t)|\boldsymbol{Q}(t)] + \xi.$$

We also have in view of (9):

$$\sum_f \boldsymbol{Q}^f(t)^T \boldsymbol{\lambda}^f \leq (1-\delta)\gamma \sum_{i\in\mathcal{I}} \pi_i \mathbb{E}[W^\star(t)|\boldsymbol{Q}(t), \mathcal{R}(t) = \mathcal{R}_i].$$

Finally, remarking that $W^\star(t) \geq \sum_{f,v} Q_{f,v}(t)/VF$, and taking δ as large as $d_\gamma(\boldsymbol{\lambda})$, we have:

$$\Delta L(t) \leq \Omega^2 V + 2\xi - 2\gamma d_\gamma(\boldsymbol{\lambda}) \sum_{f,v} Q_{f,v}(t). \quad (11)$$

Summing the above inequalities for $t = 1$ to T, we get:

$$\mathbb{E}\Big[L(j) - L(0)\Big] \leq j(\Omega V^2 + 2\xi)$$
$$- \frac{2\gamma d_\gamma(\boldsymbol{\lambda})}{VF} \sum_{t=0}^{j-1} \sum_{f,v} \mathbb{E}[Q_{f,v}(t)].$$

We deduce the following, which completes the proof by letting $j \to \infty$:

$$\frac{1}{j}\sum_{t=0}^{j-1} \sum_{f,v} \mathbb{E}[Q_{f,v}(t)] \leq \frac{VF(\Omega V^2 + 2\xi)}{2\gamma d_\gamma(\boldsymbol{\lambda})} + \frac{VF\mathbb{E}[L(Q(0))]}{2j\gamma d_\gamma(\boldsymbol{\lambda})}. \quad \square$$

Proof of Theorem 4.1. Same as Lemma 3.2 except that $\mathbb{E}[t_i - t_{i-1}] = m$, for $i = 0, \ldots$.

Proof of Theorem 5.1. We first let

$$L_1(t) \;=\; \sum_f \boldsymbol{Q}^f(t)^T R^f \mathbb{E}[\boldsymbol{S}^f(t)|\boldsymbol{Q}(t)],$$

$$L_2(t) \quad = \quad -\Big(\sum_{f,v} \beta U_{f,v}(t) - \sum_{f} \boldsymbol{Q}^f(t)^T \mathbb{E}\Big[\boldsymbol{A}^f(t)|\boldsymbol{Q}^f(t)\Big]\Big).$$

Using the quadratic Lyapunov function $L(t)$, we get:

$$\Delta L(t) \quad = \quad b + 2\Big(L_1(t) + L_2(t) + \beta \sum_{f,v} U_{f,v}(t)\Big). \quad (12)$$

Now, for a given $0 < \epsilon \leq \lambda_{max}$, define Λ_ϵ to be:

$$\Lambda_\epsilon \triangleq \{(\lambda_{f,v})_{f,v} \mid (\lambda_{f,v} + \epsilon)_{f,v} \in \Lambda, \lambda_{f,v} \geq 0, \forall f \in \mathcal{F}, v \in \mathcal{V}\}.$$

Let $\boldsymbol{\lambda}^\star(\epsilon) = (\lambda^\star_{f,v}(\epsilon), f \in \mathcal{F}, v \in \mathcal{V})$ to be the solution of the optimization problem: $\max_{(\lambda_{f,v}) \in \Lambda_\epsilon} U_{f,v}(\lambda_{f,v})$, and we let $\boldsymbol{\lambda}^\star = \boldsymbol{\lambda}^\star(0)$, for simplicity. Then, from the similar argument in [15], we get

$$\Delta L(t) = b + 2\xi - 2\epsilon \sum_{f,v} Q_{f,v}(t) - 2\beta\Big(\sum_{f,v} U^\star_{f,v}(\epsilon) - \sum_{f,v} U_{f,v}(t)\Big),$$

where $\boldsymbol{\epsilon} = (\epsilon)_{v \in \mathcal{V}}$, $(\boldsymbol{\lambda}^\star)^f(\epsilon) = (\lambda^\star_{f,v}(\epsilon), v \in \mathcal{V})$, and $U^\star_{f,v}(\epsilon) = U_{f,v}((\lambda^\star_{f,v}(\epsilon)))$. Using the technique in the proof of Theorem 3.2 for delay bound and convexity of $U_{f,v}(\cdot)$ and Jensen's inequality, the result follows. □

8. REFERENCES

[1] P. Chaporkar, K. Kar, and S. Sarkar. Throughput guarantees through maximal scheduling in wireless networks. In *Proc. of Allerton*, 2005.

[2] P. Chaporkar and S. Sarkar. Stable scheduling policies for maximizing throughput in generalized constrained queueing. In *Proc. of Infocom*, 2006.

[3] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, Joint optimal congestion control, routing, and scheduling in wireless ad hoc networks, In *Proc. of Infocom*, 2006.

[4] A. Eryilmaz, A. Ozdaglar, and E. Modiano. Polynomial complexity algorithms for full utilization of multi-hop wireless networks. In *Proc. of Infocom*, 2007.

[5] M. Hanckowiak, M. Karonski, and A. Panconesi. On the distributed complexity of computing maximal matchings. In *Proc. of the ninth annual ACM-SIAM symposium on discrete algorithms*, 1998.

[6] C. Joo and N. B. Shroff. Performance of random access scheduling schemes in multi-hop wireless networks. In *Proc. of Infocom*, 2007.

[7] K. Jung and D. Shah. Low delay scheduling in wireless network. In *Proc. of ISIT*, 2007.

[8] A. Kabbani, T. Salonidis, and E. W. Knightly. Distributed low-complexity maximum-throughput scheduling for wireless backhaul networks. In *Proc. of Infocom*, 2007.

[9] K. Kar, S. Sarkar, and L. Tassiulas. Achieving proportional fairness using local information in aloha networks. *IEEE Transactions on Automatic Control*, 49:1858–1863, 2004.

[10] E. Leonardi, M. Mellia, F. Neri, and M. A. Marsan. Bounds on delays and queue lengths in input-queued cell switches. *Journal of ACM*, 50(4):520–550, 2003.

[11] X. Lin and S. Rasool. Constant-time distributed scheduling policies for ad hoc wireless networks. In *Proc. of CDC*, 2006.

[12] X. Lin and N. B. Shroff. The impact of imperfect scheduling on cross-layer rate control in wireless networks. In *Proc. of Infocom*, 2005.

[13] S. P. Meyn and R. L. Tweedie. *Markov chains and stochastic stability*. Springer-Verlag, 1993.

[14] E. Modiano, D. Shah, and G. Zussman. Maximizing throughput in wireless networks via gossiping. In *Proc. of ACM Sigmetrics*, 2006.

[15] M. J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. In *Proc. of IEEE INFOCOM*, 2005.

[16] M. J. Neely, E. Modiano, and C. E. Rohrs. Tradeoffs in delay guarantees and computation complexity for nxn packet switches. In *Proc. of CISS*, 2002.

[17] M. J. Neely, E. Modiano, and C. E. Rohrs. Dynamic power allocation and routing for time varying wireless networks. In *Proc. of Infocom*, 2003.

[18] R. Preis. Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs. In *Proc. of STOC*, 1999.

[19] S. Ray and S. Sarkar. Arbitrary throughput versus complexity tradeoffs in wireless networks using graph partitioning. In *Proc. of Information Theory and Applications Second Workshop*, 2007.

[20] S. Sanghavi, L. Bui, and R. Srikant. Distributed link scheduling with constant overhead. In *Proc. of ACM Sigmetrics*, 2007.

[21] S. Sarkar and K. Kar. Achieving 2/3 throughput approximation with sequential maximal scheduling under primary internference constraints. In *Proc. of Allerton*, 2006.

[22] D. Shah and M. Kopikare. Delay bounds for approximate maximum weight matching algorithms for input queued switches. In *Proc. of Infocom*, 2002.

[23] G. Sharma, R. R. Mazumdar, and N. B. Shroff. On the complexity of scheduling in wireless networks. In *Proc. of Mobicom*, 2006.

[24] L. Tassiulas. Linear complexity algorithms for maximum throughput in radionetworks and input queued switches. In *Proc. of Infocom*, 1998.

[25] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1949, December 1992.

[26] V. V. Vazirani. *Approximate Algorithms*. Springer-Verlag, 2001.

[27] X. Wu and R. Srikant. Bounds on the capacity region of multi-hop wireless networks under distributed greedy scheduling. In *Proc. of Infocom*, 2006.

[28] Y. Yi and M. Chiang. Stochastic network utility maximization. *European Transactions on Telecommunications*, March, 2008.

[29] Y. Yi and M. Chiang. Wireless scheduling with O(1) complexity for m-hop interference model. In *Proc. of ICC*, 2008.