Practical Computation of Optimal Schedules in Multihop Wireless Networks

Peng Wang Stephan Bohacek
pwangee@udel.edu bohacek@udel.edu
Department of Electrical and Computer Engineering
University of Delaware

Abstract—**Interference and collisions greatly limit the throughput of mesh networks that use contention-based MAC protocols such as 802.11. Significantly higher throughput is achievable if transmissions are scheduled. However, traditional methods to compute optimal schedules are computationally intractable (unless co-channel interference is neglected). This paper presents a practical technique to compute optimal schedules. The resulting algorithm searches for a low dimensional optimization problem that has the same solution as the full problem. Such a low dimensional problem is shown to always exist. The resulting algorithm converges arithmetically fast or geometrically fast, depending on whether the objective is to maximize the proportional fair throughput or to maximize the minimum throughput, where the minimum is over all flows in the network. At each iteration of the algorithm, a graph theoretic optimization known as the maximum weighted independent set (MWIS) problem must be solved. While the general MWIS problem is NP-complete in the worst-case, we find that the MWIS can be solved efficiently, e.g., on current computers it can be solved in about 1 sec on a network with 2048 nodes. Finally, the impact of optimal scheduling is examined on realistic models of mesh networks, where it is found that the throughput provided by optimal scheduling is between a factor of 3 and 11 greater than that provided by 802.11's CSMA/CA, motivating further research in optimal scheduling.**

I. Introduction

802.11-based mesh networks are being deployed or planning to be deployed in over 300 cities [1]. One motivation for 802.11-based mesh networks is that mesh routers can be densely deployed with relatively low cost. A dense deployment of routers results in the typical mobile user being close to at least one router, allowing high data rate communication between the user and the router. However, a dense distribution of routers also results in significant interference. Due to the poor performance of CSMA/CA in environments with high interference, it is unclear if 802.11 with CSMA/CA will provide sufficient data rates for future mobile applications. An alternative to CSMA/CA is to schedule some fraction of the transmissions.

Achieving high throughput in the face of interference has been an active area of research for at least 25 years [2]. However, nearly 20 years ago it was shown that computing optimal schedules is *potentially* NP-complete [3]. On the other hand, it has never been shown that the cases where the throughput maximization is NP-complete actually arise in wireless networks. Notably, it has been proved that if there is no co-channel interference, then optimal schedules can be computed in polynomial time [4]. Co-channel interference arises when two nodes transmit simultaneously and, due to the interference, impede the ability of the receivers to correctly decode the messages. Under the assumption that co-channel interference does not arise, tremendous progress has been made (e.g., [4]–[6], [9], [10], [13]). Unfortunately, with the dense deployment of mesh routers, co-channel interference is expected to be significant. Moreover, schedules generated under the assumption that co-channel interference does not arise, tend to perform quite poorly when there is co-channel interference (e.g., [15]).

This paper presents practical techniques for computing optimal schedules even when co-channel interference exists. The ability to compute the schedules of a 2048 node network densely covering downtown Chicago is demonstrated. There are two key theoretical results that underpin this approach.

- Letting L be the number of links in the network, a brute-force approach requires optimization over a space with 2^L elements. However, the optimal solution requires no more than L elements. If these L special elements were somehow known in advance, then the optimization could be performed over a space with L elements and the result would be identical to the one found by optimizing over the space of all elements.
- From a solution of the optimization problem over an arbitrary set of L elements, either
  - a new set of elements can be found that will improve the solution, which, in turn, leads to a better set of elements, and so on,
  - or, if no set of better elements exists, then the current set of elements is optimal.

These observations lead to an algorithm that converges either arithmetically or geometrically, when the objective is to maximize the proportionally fair throughput or the minimum throughput, respectively. However, at each iteration a graph theoretic problem known as the maximum weighted independent set (MWIS) problem must be solved. The MWIS problem on general graphs is NP-complete, however, as mentioned above, the MWIS problem is not necessarily NP-complete on graphs associated with scheduling on wireless networks [16]. This paper demonstrates that the MWIS problem can be solved in about 1 sec for a practical network with 2048 nodes.

As an example, this paper considers the achievable throughput of a realistic model of a mesh network in downtown Chicago, where the model was developed with the UDel Models urban network simulator [17]. When compared to 802.11 with CSMA/CA, optimal scheduling typically increases the throughput by a factor of 3 to 11, depending on the density of the wired gateways. Such large improvements in throughput justify further research in optimal scheduling.

The remainder of the paper proceeds as follows. In the next section, the system model, notation, and problem definition are given. Optimal scheduling is discussed in Section III. Results from numerical experiments are presented in Section IV. Concluding remarks are given in Section V. All proofs can be found in the Appendix.

II. System Model and Problem Formulation

A router-to-router flow is denoted by ϕ, with Φ denoting the set of all such flows. To improve presentation, it is assumed that all flows use a single path, however, the extension to multipath is straightforward. The data rate of flow ϕ is denoted f_ϕ, and the path followed by flow ϕ is denoted $P(\phi)$. The set of all considered paths is $\mathcal{P}$. Using this notation, the total data rate sent over link x is $\sum_{\{\phi | x \in P(\phi)\}} f_\phi$, where $\{\phi | x \in P(\phi)\}$ is the set of flows that cross link x. All links are directional.

We define an *assignment* to be a vector $\mathbf{v} = \left[\begin{array}{ccc} v_1 & \cdots & v_L \end{array} \right]$, where there are L links in the network and where $v_x \in \{0,1\}$ with $v_x = 1$ implying that link x is active during assignment $\mathbf{v}$. It is possible to extend this approach to accommodate links with multiple bit-rates and multiple transmit powers. The *set of considered assignments* is denoted by $\mathcal{V}$, while the *set of all assignments* is denoted by $\overline{\mathcal{V}}$. In this simple case where $v_x \in \{0,1\}$, $\overline{\mathcal{V}}$ has 2^L assignments. The size of $\overline{\mathcal{V}}$ is the main challenge facing optimal scheduling. Thus, typically, we only consider a subset of all assignments, i.e., $\mathcal{V} \subsetneq \overline{\mathcal{V}}$.

The data rate across link x during assignment $\mathbf{v}$ is denoted by $R(\mathbf{v}, x)$. In general $R(\mathbf{v}, x)$ is a complicated function. However, here a simple binary relationship is used to define $R(\mathbf{v}, x)$. Specifically,

$$R(\mathbf{v}, x) = \left\{ \begin{array}{l} R_x \text{ if } v_y = 0 \text{ for all } y \in \chi(x) \\ 0 \text{ otherwise} \end{array} \right. , \tag{1}$$

where $\chi(x)$ is a set of links that conflict with x, i.e., $y \in \chi(x)$ if simultaneous transmissions over x and y are not possible. R_x is the *nominal data rate* over link x. Evaluation of techniques to select the nominal data rate for a link is outside the scope of this paper. For this paper, let R_x be the maximum data rate that the SNR across link x can support. Note that this definition of $R(\mathbf{v}, x)$ neglects the possibility of transmission errors due to the aggregate interference from several links not in $\chi(x)$. However, as discussed in Section III-F, such problems can easily be addressed. All computations in this paper use this technique, and hence the computed capacities account for multiple interferers. This definition of $R(\mathbf{v}, x)$ also neglects the possibility that $R(\mathbf{v}, x)$ can take intermediate values between R_x and 0. For example, in the face of moderate interference, retransmissions will result in an effective data rate that is below R_x. With a slight modification, this behavior can be supported by employing a multi-valued definition of $R(\mathbf{v}, x)$. Future work will investigate such modifications.

The set of conflicting link, $\chi(x)$, depends on the communication model. Arguably, the $SINR$ protocol communication model is the most relevant and is the model used in this paper. Let $SINR(x, y)$ be the SINR at the receiver of link x when link y is also active. That is, $SINR(x, y) := H_{x,x} / (H_{y,x} + \mathcal{N})$ where $H_{x,x}$ is the strength of the signal transmitted from the transmitter of link x to the receiver of link x, $H_{y,x}$ is the strength of the signal transmitted from the transmitter of link y to the receiver of link x, and $\mathcal{N}$ is the strength of the noise. Then, the SINR communication model specifies that $y \in \chi(x)$ if $SINR(x, y) < T(x)$ or $SINR(y, x) < T(y)$, where $T(x)$ and $T(y)$ are thresholds that depend on the modulation schemes.

A schedule is a convex combination of assignments. Specifically, a schedule is a set $\{\alpha_\mathbf{v} : \mathbf{v} \in \mathcal{V}\}$ where $\sum_{\mathbf{v} \in \mathcal{V}} \alpha_\mathbf{v} \le 1$ and $\alpha_\mathbf{v} \ge 0$. Thus, $\alpha_\mathbf{v}$ is the fraction of time that assignment $\mathbf{v}$ is used. With this notation, the total data rate that the schedule $\boldsymbol{\alpha}$ provides over link x is $\sum_{\mathbf{v} \in \mathcal{V}} \alpha_\mathbf{v} R_x v_x$. Finally, the throughput optimization problem is

$$\max_{\boldsymbol{\alpha}, \mathbf{f}} G(\mathbf{f}) \tag{2a}$$

subject to:

$$\sum_{\{\phi | x \in P(\phi)\}} f_\phi \le \sum_{\mathbf{v} \in \mathcal{V}} \alpha_\mathbf{v} R(\mathbf{v}, x) \text{ for each link } x \tag{2b}$$

$$\sum_{\mathbf{v} \in \mathcal{V}} \alpha_\mathbf{v} \le 1 \tag{2c}$$

$$0 \le \alpha_\mathbf{v} \text{ for each } \mathbf{v} \in \mathcal{V}, \tag{2d}$$

where $\mathbf{f}$ is the vector of flow rates. The function G is referred to as the *throughput metric*. Several different throughput metrics are possible. In some cases, the throughput metric is the network utility $G(\mathbf{f}) = \sum_{\phi \in \Phi} U_\phi(f_\phi)$, where U_ϕ is the utility function for flow ϕ. Popular utility functions include $U_\phi(f) = w_\phi \log(f)$ [18]–[20] and $U_\phi(f) = w_\phi f^{1-\gamma} / (1-\gamma)$ [21], where w_ϕ is the administrative weight. Another widely used throughput metric is $G(\mathbf{f}) = \min_{\phi \in \Phi} w_\phi f_\phi$ [22]. This paper specifically, focuses on the cases when $G(\mathbf{f}) = \sum_{\phi \in \Phi} w_\phi \log(f_\phi)$ and $G(\mathbf{f}) = \min_{\phi \in \Phi} w_\phi f_\phi$. In the case that $G(\mathbf{f}) = \sum_{\phi \in \Phi} w_\phi \log(f_\phi)$, the objective function is continuously differentiable, concave, and increasing. The solvability of such problems is detailed in [23]. If $G(\mathbf{f}) = \min_{\phi \in \Phi} w_\phi f_\phi$, then (2) can be written as a linear programming problem, which is extensively studied in [24].

In theory, (2) is solvable. However, there is a significant computational challenge in that if $\mathcal{V}$ is the set of all assignments, then the vector $\boldsymbol{\alpha}$ has 2^L elements. Thus, the size of the space over which the optimization is performed must be reduced. This idea of considering a reduced space was considered in [22] and [25], however, the space was constructed arbitrarily. In this paper the space is constructed so that the throughput found by optimizing over the reduced space is the same throughput found by optimizing over the entire space.

III. Optimal Scheduling

A. Introduction

The objective of this section is to compute optimal schedules by optimizing over a set of considered assignment $\mathcal{V} \subsetneq \overline{\mathcal{V}}$. The key questions are 1). is it possible to reduce the size of $\mathcal{V}$ without impacting the solution, and 2). if so, how can the set of considered assignments be constructed so that the value of (2) with the reduced sized $\mathcal{V}$ is the same or near to the value when $\mathcal{V} = \overline{\mathcal{V}}$? The answer to the first question is provided next and the following subsections focus on the second question.

Theorem 1: There exists $\mathcal{V}$ with L assignments such that the solution to (2) is the same as the solution to (2) when $\mathcal{V} = \overline{\mathcal{V}}$.

The full proof of this theorem is in Section VI-A. The result, which follows from Caratheodory's Theorem (e.g., Theorem B.6 in [23]), implies that the optimal schedule can be found by considering a set, $\mathcal{V}$, that is relatively small.

B. Basics

It is well known that Lagrange multiplier theory can be applied to (2) (e.g., see [23]). Specifically, associated with each link constraint (2b) is a Lagrange multiplier denoted by μ_x, with $\boldsymbol{\mu}$ being the vector of such multipliers. Similarly, associated with the constraint (2c) is a Lagrange multiplier denoted by λ. Employing the economic interpretation of Lagrange multipliers, μ_x can be interpreted as the price/bit of sending data over links x, or from the network's point of view, μ_x is the revenue that is collected for each bit that crosses link x. Under this interpretation, the revenue generated by assignment $\mathbf{v}$ is

$$\sum_{x=1}^{L} R(\mathbf{v}, x)\, \mu_x.$$

The multiplier λ can be interpreted as the maximum revenue generated by any assignment in $\mathcal{V}$. Specifically,

Theorem 2: Let $G(\mathbf{f}) = \sum_{\phi \in \Phi} w_\phi \log(f_\phi)$ or $G(\mathbf{f}) = \min_{\phi \in \Phi} w_\phi f_\phi$ and let μ_x be the Lagrange multiplier associated with constraint (2b) and let λ be the Lagrange multiplier associated with constraint (2c), then

$$\lambda = \max_{\mathbf{v} \in \mathcal{V}} \sum_{x=1}^{L} R(\mathbf{v}, x)\, \mu_x. \tag{3}$$

The proof of this proposition is in Section VI-A.

Typically, there are many assignments in $\mathcal{V}$ that generate revenue λ. The set of such assignments is referred to as the set of active assignments and is denoted $\mathcal{V}^*$, i.e.,

$$\mathcal{V}^*(\boldsymbol{\mu}) := \left\{ \mathbf{v} : \sum_{x=1}^{L} R(\mathbf{v}, x)\, \mu_x = \max_{\mathbf{v} \in \mathcal{V}} \sum_{x=1}^{L} R(\mathbf{v}, x)\, \mu_x \right\}. \tag{4}$$

From Theorem 1, the optimal schedule multiplexes between a set of no more than L assignments. These assignments are contained in the set $\mathcal{V}^*$.

Theorem 3: If $\mathbf{v} \notin \mathcal{V}^*$, then $\alpha_{\mathbf{v}} = 0$.

The proof of this proposition is in Section VI-A.

Since the revenue generated by active assignments is λ, the optimal schedule also generates revenue λ. Specifically, let R_x^* be the optimal data rate across link x, that is

$$R_x^* := \sum_{\mathbf{v} \in \mathcal{V}} \alpha_{\mathbf{v}}^* R(\mathbf{v}, x), \tag{5}$$

where α^* specifies the optimal schedule. Then, it can easily be shown that

$$\lambda = \sum_{x=1}^{L} R_x^* \mu_x.$$

C. Evaluating Candidate Assignments

A brute force approach to construct a good set of assignments is to start with an arbitrary set of assignments, $\mathcal{V}$, select an assignment $\mathbf{v}^+ \notin \mathcal{V}$, and evaluate the resulting throughput with the set of assignments $\mathbf{v}^+ \cup \mathcal{V}$. However, this approach is computationally complex in that (2) must be repeatedly solved. Furthermore, it is not clear if the utility of $\mathbf{v}^+$ is only apparent when it is added to $\mathcal{V}$ along with a particular set of other assignments. Alternatively, the question of whether an assignment $\mathbf{v}^+ \notin \mathcal{V}$ will increase the throughput when the set of considered assignments is changed from $\mathcal{V}$ to $\mathbf{v}^+ \cup \mathcal{V}$ is answered by the following theorem.

Theorem 4: For the set of assignments $\mathcal{V}$, let $\boldsymbol{\mu}$ and λ be the Lagrange multipliers associated with constraints (2b) and (2c) when (2) is solved with this $\mathcal{V}$. Now consider an assignment $\mathbf{v}^+ \notin \mathcal{V}$, The throughput provided by $\mathbf{v}^+ \cup \mathcal{V}$ is greater than that provided by $\mathcal{V}$ if and only if

$$\sum_{x=1}^{L} R(\mathbf{v}^+, x)\, \mu_x - \lambda > 0. \tag{6}$$

Corollary 5: If Lagrange multipliers that result from optimizing over $\mathcal{V}$ are such that there does not exist an assignment that satisfies (6), then the schedule found by optimizing over $\mathcal{V}$ is optimal.

The proof of Theorem 4 is in Section VI-A.

Theorem 4 provides the main tool for constructing a good set of assignments. Invoking an economic interpretation of the Lagrange multipliers, Theorem 4 implies that an assignment $\mathbf{v}^+$ will increase the utility if it generates more revenue per second than any other assignments in the set $\mathcal{V}$.

D. Algorithm to Maximize the Throughput

Based on Theorem 4, Algorithm 1 can be used to iteratively add assignments to $\mathcal{V}$ such that the added assignment satisfies (6). The intuition behind this algorithm is to compute the values of the Lagrange multipliers by solving (2) with $\mathcal{V} = \mathcal{V}(n)$, where $\mathcal{V}(n)$ is the set of assignments at the nth iteration. With these multipliers a new assignment is found that satisfies (6). If no such assignment exists, then Corollary 5 implies that the schedule is optimal. If an assignment that satisfies (6) is found, then we set to $\mathcal{V}(n+1)$ the union of $\mathcal{V}(n)$ and this newly found assignment. With this new set of assignments, (2) is resolved and an improvement of the resulting throughput is guaranteed by Theorem 4. This process is repeated until no new assignments can be found (in which case the optimal schedule has been found) or until the current solution is close enough to optimal.

The first step of Algorithm 1 requires an initial set of assignments, $\mathcal{V}(0)$. This initial set of assignments must result in a solution to (2) where the flow rates are non-zero. The initial set of assignments can be found using a greedy approach given by Algorithm 2. We note that a wide range of techniques could be employed to select $\mathcal{V}(0)$. An examination of the performance of these various techniques is left for future work.

The following indicates the convergence of Algorithm 1.

Theorem 6: Let $\{(\mathbf{f}(n), \boldsymbol{\alpha}(n)) | n = 1, 2, ...\}$ be the sequence of solutions given by Algorithm 1 with $\rho = 0$. Then $\lim_{n\to\infty}(\mathbf{f}(n), \boldsymbol{\alpha}(n)) = (\mathbf{f}(\infty), \boldsymbol{\alpha}(\infty))$, the optimal solution of (2) when $\mathcal{V} = \overline{\mathcal{V}}$.

Theorem 7: If $G(\mathbf{f}) = \min_{\phi\in\Phi} f_\phi$, then $(G(\mathbf{f}(\infty)) - G(\mathbf{f}(n+1))) / (G(\mathbf{f}(\infty)) - G(\mathbf{f}(n))) < \delta$ for some constant $0 < \delta < 1$, that is, $G(\mathbf{f}(n))$ converges to $G(\mathbf{f}(\infty))$ geometrically fast.

Theorem 8: Let $A \in \{0,1\}^{|\Phi|\times L}$ with $A(\phi, x) = 1$ if $x \in P(\phi)$ and $A(\phi, x) = 0$ otherwise. Suppose that the null space of A is empty. Then Algorithm 1 with $\rho = 0$ converges arithmetically when $G(\mathbf{f}) = \sum_{\phi\in\Phi}\log(f_\phi)$, that is, $G(\mathbf{f}(\infty)) - G(\mathbf{f}(n)) \leq a/n$ for some $a > 0$.

The proofs of these three theorems are in Sections VI-B, VI-C, and VI-D, respectively.

The condition that A has an empty null space is satisfied if no link has the exact same set of flows crossing it. In the case that the same set of flows do cross two different links, it is often the case that only one of these links will be a bottleneck and hence μ_x will be zero for the non-bottleneck link. Under this restriction, it is straightforward to show that the conclusion of Theorem 8 holds.

Theorem 9: Suppose that Algorithm 1 terminates after n^* iterations with $\rho > 0$. If $G(\mathbf{f}) = \min_{\phi\in\Phi} f_\phi$, then $\frac{G(\mathbf{f}(\infty)) - G(\mathbf{f}(n^*))}{G(\mathbf{f}(\infty))} < \rho$. Whereas, if $G(\mathbf{f}) = \sum_{\phi\in\Phi}\log(f_\phi)$, then $G(\mathbf{f}(\infty)) - G(\mathbf{f}(n^*)) = \sum_{\phi\in\Phi}\log(f_\phi(\infty)/f_\phi(n^*)) < L\log(1+\rho)$.

This theorem is proved in Section VI-C.

In our computational experiments we found that Algorithm 1 suffers from numerical issues that reduce the rate of convergence when n and L are large. It appears that solving (2) when $G(\mathbf{f}) = \sum_{\phi\in\Phi}\log(f_\phi)$ is the source of these numerical problems. Thus, when L is large and $G(\mathbf{f}) = \sum_{\phi\in\Phi}\log(f_\phi)$, we recommend $\rho > 0$. Specifically, when $L \leq 512$, we have found that $\rho = 0.05$ works well, but for $L > 512$, we use $\rho = 0.15$. It is hoped that improvements in solving (2) for large L will allow smaller values of ρ.

E. Finding New Assignments

Algorithm 1 changes the challenge of solving (2) over a large set of assignments to the challenge of finding assignments that solves (6). More specifically, Algorithm 1 requires solving

$$\max_{\mathbf{v}\in\overline{\mathcal{V}}} \sum_{x=1}^{L} R(\mathbf{v}, x)\,\mu_x. \qquad (8)$$

Algorithm 1 Computing an Optimal Schedule

```
1:  Select an initial set of assignments $\mathcal{V}(0)$, select a level of
    accuracy $\rho \geq 0$, and set $n = 0$.
2:  Solve (2) with $\mathcal{V} = \mathcal{V}(n)$, and, hence, compute the flow
    rates $\mathbf{f}(n)$ and the Lagrange multipliers $\boldsymbol{\mu}(n)$ and $\lambda(n)$
    associated with constraints (2b) and (2c), respectively.
3:  Search for an assignment $\mathbf{v}(n)$ such that
        $\mathbf{v}(n) = \arg\max_{\mathbf{v}\in\overline{\mathcal{V}}} \sum_{x=1}^{L} R(\mathbf{v}, x)\,\mu_x(n) - \lambda(n).$   (7)
4:  case $G(\mathbf{f}) = min_{\phi\in\Phi} f_\phi$
5:      if $\frac{\sum_{x=1}^{L} R(\mathbf{v}(n),x)\mu_x(n) - \lambda(n)}{G(\mathbf{f}(n))} < \rho$
6:          Stop.
7:      end if
8:  case $G(\mathbf{f}) = \sum_{\phi\in\Phi}\log(f_\phi)$
9:      if $\sum_{x=1}^{L} R(\mathbf{v}(n), x)\,\mu_x(n) - \lambda(n) < L\log(1+\rho)$
10:         Stop.
11:     end if
12: set $\mathcal{V}(n+1) = \mathcal{V}(n) \cup \mathbf{v}(n)$
13: set $n = n + 1$, and go to Step 2.
```

Algorithm 2 Selecting an Initial Set of Assignments

```
1:  Set $\mathcal{V}$ empty, and set $w_x = 0$ for all links $x$.
2:  Start an assignment $\mathbf{v}$ with $v_x = 0$ for all $x$.
3:  Randomly select a link $x$ such that $w_x = 0$. Set $w_x = 1$
    and $v_x = 1$.
4:  Randomly select a link $y$ such that $w_y = 0$ and $y \notin \bigcup_{\{x|v_x=1\}} \chi(x)$.
    Furthermore, check whether each active
    link in assignment $\mathbf{v}$ satisfy the desired SINR requirement
    (see Section III-F).
5:  if such a $y$ exists then
        set $w_y = 1$, $v_y = 1$ and go to Step 4.
6:  else
        set $\mathcal{V} = \mathcal{V} \cup \mathbf{v}$.
7:  end if
8:  if for all $x$ there exits a $\mathbf{v} \in \mathcal{V}$ such that $v_x = 1$ then
        stop. $\mathcal{V}$ is the set of initial assignments.
9:  else
        go to Step 2
10: end if
```

As will be shown next, solving this maximization is equivalent to finding the maximum weighted independent set of the weighted conflict graph.

The utility of the conflict graph for finding schedules has been demonstrated in several previous works (e.g., [3], [22]). The conflict graph is constructed as follows. Each link in the network induces a vertex in the conflict graph. Thus, a link x in the network is associated with a vertex in the conflict graph; this vertex is denoted with x, where whether x refers to a link in the network or a vertex in the conflict graph is clear from the context. There is an edge between vertices x and y if $y \in \chi(x)$, where, as discussed in Section II, x and links in $\chi(x)$ cannot simultaneously active. The weighted conflict graph is constructed by assigning the weight $R_x \mu_x$ to vertex x, where R_x is the nominal data rate across link x and μ_x is the Lagrange multiplier associated with constraint (2b).

An independent set (or stable set) of a graph is a set of vertices where no two vertices in the set are neighbors. Letting I be an independent set, the weight of I is the sum of the weights of the vertices in I. Thus, an independent set of the conflict graph is a set of links that are not in conflict and hence able to active simultaneously. Therefore, if I is an independent set and $\mathbf{v}(I)$ is the assignment generated by I via $v_x(I) = 1$ if $x \in I$, then under assignment $\mathbf{v}(I)$ the data rate across link x is R_x. Furthermore, the weight of I is $\sum_{x \in I} R_x \mu_x$. By (1), $\sum_{x \in I} R_x \mu_x = \sum_{x=1}^{L} R(\mathbf{v}, x) \mu_x$. Thus, the goal of solving (8) is the same as finding the maximum weighted independent set (MWIS).

Unfortunately, for general graphs, finding the MWIS is NP-complete. On the other hand, the MWIS problem has been extensively studied. For example, it is known to be solvable in polynomial time for many classes of networks including perfect graphs [26], interval graphs (which arise when a wireless network is confined to a road) [26], disk graphs [27], claw-free graphs [28], fork-free graphs [29], trees [30], sparse random graphs [31], and circle graphs [32]. Moreover, there has been extensive work on approximating the MWIS (see [33] for a review) and specialized algorithms have been developed for exactly computing the MWIS [34], [35], [36], [40]. However, after evaluating several alternative approaches, we have found the MWIS can be quickly computed with a generic integer linear programming (ILP) solver. The MWIS problem can be formulated as an ILP via

$$\max_{\mathbf{v}} \sum_{x=1}^{L} R_x \mu_x v_x \tag{9}$$

$$\text{subject to: } v_x + v_y \leq 1 \text{ if } y \in \chi(x) \tag{10}$$

$$v_x \in \{0, 1\}.$$

In large networks, there are many constraints (10). The computation time can be dramatically improved if a clique[1] decomposition is used, where we define a clique decomposition to be a set of cliques $\{Q_i, \, i = 1, 2, ...M\}$ such that if $y \in \chi(x)$, then there is a clique Q_i such that $x \in Q_i$ and $y \in Q_i$. Then, (9) becomes

$$\max_{\mathbf{v}} \sum_{x=1}^{L} R_x \mu_x v_x \tag{11}$$

$$\text{subject to: } \sum_{x \in Q_i} v_x \leq 1 \text{ for } i = 1, 2, ..., M$$

$$v_x \in \{0, 1\}.$$

While an optimal clique decomposition might further improve the computation time, we have found that a simple greedy clique decomposition results in a factor of ten improvement over (9). Section IV shows that (11) can be solved in about 1 sec for networks with 2048 nodes (or 1984 links).

[1]A clique is a set of vertices where there is an edge between each vertex in the set .

Remark 10: This paper focuses on the SINR binary-conflict communication model. If co-channel interference does not arise, then the *node exclusive model* can be used. Under the node exclusive model, $y \in \chi(x)$ if link's x transmitter or receiver are the same as y's transmitter or receiver. In this case, the conflict graph is a line-graph, and the MWIS problem reduces to a maximum weighted matching (WMM) problem [26]. In the general case, polynomial complexity algorithms exist for solving the WMM problem (e.g., see [41] for an $O\left(N \cdot L + N^2 \log(N)\right)$ complexity algorithm where N is the number of nodes and L is the number of links). In the case of a tree (which is typical for today's mesh networks), optimal matching is possible with simple greedy algorithms and message passing algorithms [42]. In the case of general topologies, simple approximation is also possible since maximal weighted matching is within a factor of two of optimal [43]. There exist distributed algorithms with approximation ratios of $1/2$ [44], [45] and $2/3$ [46]. However, since the node exclusive model neglects co-channel interference, the schedules produced under this model perform poorly when applied to actual networks where co-channel interference exists [15].

F. Correcting Multi-Conflicts

The model (1) is a binary model in that it only considers conflicts between two links. However, conflicts between more than two links can occur. For example, it is possible that $x \notin \chi(y)$, $x \notin \chi(z)$, and $y \notin \chi(z)$. Thus, according to the binary conflict model, links x, y, and z can all simultaneously active. However, it is possible that the combined interference from y and z, results in enough interference such that transmission across link x fails with high probability. In this case, we say that the links x, y, and z form a multi-conflict. Schedules that use assignments that contain multi-conflicts will have low throughput when deployed. Thus, such assignments should be removed. While the scheme described above removes all binary conflicts, as described next, we remove multi-conflicts only as they arise.

Let $\mathbf{v}^+$ be an assignment found by solving (8). $\mathbf{v}^+$ has a multi-conflict if there is a link x with $v_x^+ = 1$ and links $\{y_i : i = 1, 2, ...K\}$ with $v_{y_i}^+ = 1$ and

$$T(x) > SINR(x, \{y_1, y_2, ..., y_K\}) := \frac{H_{x,x}}{\sum_{i=1}^{K} H_{y_i,x} + \mathcal{N}}. \tag{12}$$

This multi-conflict is defined by the set $C = \{x\} \cup \bigcup_{i=1}^{K} \{y_i\}$. An assignment that maximizes (8) and yet does not contain this multi-conflict can be found by solving

$$\max_{\mathbf{v}} \sum_{x=1}^{L} R_x \mu_x v_x \tag{13}$$

$$\text{subject to: } \sum_{x \in Q_i} v_x \leq 1 \text{ for } i = 1, 2, ..., M$$

$$\sum_{x \in C} v_x \leq |C| - 1$$

$$v_x \in \{0, 1\},$$

where $|C|$ is the number of links in the set C. Intuitively, C should be the smallest set that contains the set of links that forms a multi-conflict at link x. Solving (13) will result in another assignment. If this assignment also has a multi-conflict, then the above problem is further modified. Thus, after N multi-conflicts are found, new assignments are found by solving

$$\max_{v} \sum_{x=1}^{L} R_x \mu_x v_x \tag{14}$$

$$\begin{aligned} \text{subject to: } & \sum_{x \in Q_i} v_x \leq 1 \text{ for } i = 1, 2, ..., M \\ & \sum_{x \in C_i} v_x \leq |C_i| - 1 \text{ for } i = 1, 2, ..., N \\ & v_x \in \{0, 1\}, \end{aligned}$$

where C_i is the ith multi-conflict.

Note that each time a multi-conflict is found, (14) must be resolved. Thus, a large number of multi-conflicts can result in significant computation. The next section finds that only a small number of multi-conflicts arise when forming schedules in practical mesh networks. Also, note that it is important that the initial set of assignments constructed with Algorithm 2 is free from multi-conflicts.

IV. Numerical Experiments in Optimal Scheduling

A. Experimental Set-Up

As discussed above, determining the optimal throughput has a theoretical worst-case computational complexity that makes computing throughput intractable for even small networks. However, the theoretical worst-case performance provides little insight into the typical computational complexity that occurs in mesh networks. Thus, it is imperative that the computational complexity be examined in realistic mesh networks. To this end, the UDel Models [17] were employed. Along with a realistic mobility simulator, the UDel Models include a map builder, a realistic propagation simulator, and large collection of data and trace files. The propagation simulator is based on ray-tracing and accounts for reflections off of the ground and off of buildings, transmission through building walls, and diffraction around and over buildings [47]. It also accounts for the impact that different materials have on reflections off of walls and transmission through walls. Data sets for several urban areas are available online.

For this investigation, two types of mesh network topologies were investigated. One class of topologies was generated from 6×6 city block regions of downtown Chicago with lamppost-mounted radios. In this case, the UDel Models were used to determine the signal strength between nodes. The 6×6 city block regions were randomly located in the 2 km^2 region. Various nodes densities were investigated. Specifically, the number of gateways[2] was 1, 2, 3, 6 and the number of wireless routers was 18, 36, 54, 72, and 90. Ten samples were made for each number of gateways and wireless routers (200 topologies in total). In these experiments, all traffic flowed from the gateways to destinations (i.e., downstream traffic), where each mesh router in the topology was a destination of a flow. The routing was a least hop routing, where each link had a SNR of at least 17.5 dB. Among paths with the same number of hops, the path selected was the one that had the highest minimum link SNR, where the minimization is over each hop along the path. Each flow originates at the gateway such that the best route from the gateway to the destination of the flow is no worse than any route from any other gateway in terms of the minimum SNR along the route. These topologies are used in Section IV-C.

A second type of topologies are used in Section IV-B. These topologies included the outdoor lamppost-mounted nodes along with indoor infrastructure nodes. In total, 7000 nodes were placed in the city. From these nodes, a wide range of topologies can be formed by selecting nodes subject to various conditions. Here, nodes were selected so that the network was connected and so that each node had approximately six neighboring nodes with which it can communicate at 24 Mbps using 802.11a. This node density resulted in the conflict graph having a degree of between 15 and 20. Once the nodes were selected, a set of gateways was selected so that the number of gateways equals the number of nodes divided by 32. The gateways were selected such that they were uniformly distributed. Finally, the routing was formed by solving a max-flow problem that also considers an approximation of interference. This routing algorithm is similar to the one presented in [6]. For further details on forming the topologies see [16]. In this way, topologies were made with 64, 128, 256, 512, 768, 1024, and 2048 nodes. In order to further investigate the computational complexity, Section IV-B2 also uses a similar set of topologies, but where the initial set of nodes were uniformly distributed and either the two-ray propagation model[3] or the two-ray with lognormal shadowing propagation model is used. The lognormal shadowing used a standard deviation of 4 dB [48]. For each number of nodes and propagation model, 40 sample topologies were generated. In all, 840 of this type of topologies were generated.

Note that since we only communicate between gateways and wireless routers, the topologies are forests (i.e., sets of trees). Thus, if N is the total number of nodes, and G is the number of gateways, then $L = N - G$ is the number of links.

Finally, 802.11a data rates were used. Specifically, for each link, the propagation model was used to determine the SNR[4]. We selected the nominal data rate so that the 802.11a physical layer is able to successfully transmit a 1000B packet with this SNR with probability 0.99. The specific relationship between SNR and bit-rate used was

$$\begin{gathered} \text{6Mbps} \Longleftrightarrow 2.5 \text{ dB; 12Mbps} \Longleftrightarrow 5.5 \text{ dB;} \\ \text{18Mbps} \Longleftrightarrow 8.5 \text{ dB; 24Mbps} \Longleftrightarrow 11.5 \text{ dB;} \\ \text{36Mbps} \Longleftrightarrow 14.5 \text{ dB; 48Mbps} \Longleftrightarrow 18.5 \text{ dB;} \\ \text{54Mbps} \Longleftrightarrow 20.5 \text{ dB.} \end{gathered}$$

[2]Gateways are nodes that have wired and wireless interfaces and provide a connection between the wireless mesh and the Internet.

[3]We assumed that the antennas were 1.5 meters above the ground. Thus, the channel gain is K/d^2 for $d < 200m$ and $K200^2/d^4$ for $d \geq 200m$, where $K = (\lambda/(4\pi))^2$ and $\lambda = 5.8$ cm, which is the wavelength at 5.13 GHz.

[4]We assumed 18dBm transmit power and a noise floor of -92.5 dBm, when the noise factor is included.

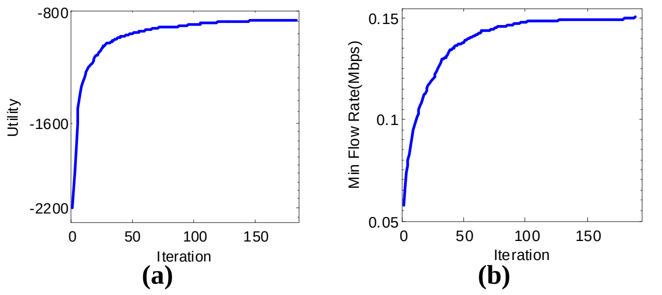

Fig. 1. Variation in the computed capacity as assignments are added. In (a) the capacity is the total utility, i.e., $\sum_{\phi\in\Phi}\log(f_\phi)$. In (b) the capacity is $\min_{\phi\in\Phi} f_\phi$. These plots are for a 1024 node (992 link) topology.

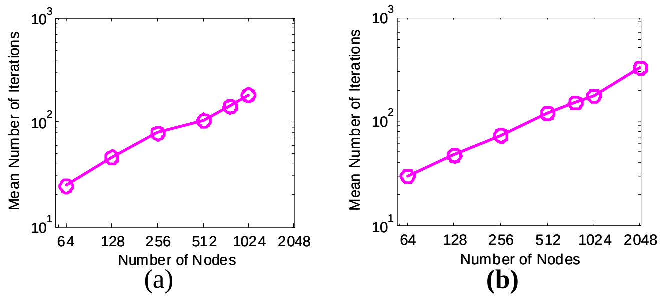

Fig. 2. Number of iterations until Algorithm 1 stopped. In (a) $G(\mathbf{f}) = \sum_{\phi\in\Phi}\log(f_\phi)$ and $\rho = 0.15$ In (b) $G(\mathbf{f}) = \min_{\phi\in\Phi} f_\phi$ and $\rho = 0.05$.

We assume that the above relationship between SNR and bit-rate also holds for SINR, e.g., if $SINR = 11.5$ dB, then a data rate of 24Mbps results in a packet success probability of 0.99.

Clearly, there are wide range of parameters used to define the topology and the bit-rates. An investigation of the computational complexity and computed throughput as a function of these parameters is currently under way.

B. Results from Numerical Experiments

1) Number of Iterations until Algorithm 1 Stops: Figure 1 shows how, in the 1024 node (992 link) topology, the throughput increases as the more assignments are added. The point of maximum throughput occurs when the solution to the ILP (11) does not satisfy (6) or the stopping condition specified in Algorithm 1 is met. Thus, in this case, Algorithm 1 stopped after 186 iterations when the throughput metric was $G(\mathbf{f}) = \sum_{\phi\in\Phi}\log(f_\phi)$, and after 191 iterations when the throughput metric was $G(\mathbf{f}) = \min_{\phi\in\Phi} f_\phi$. When $G(\mathbf{f}) = \sum_{\phi\in\Phi}\log(f_\phi)$, the stopping condition used $\rho = 0.15$, while for the case of $G(\mathbf{f}) = \min_{\phi\in\Phi} f_\phi$ we used $\rho = 0.05$. Note that since the objective functions are different, the values of ρ should not be compared.

As can be observed, the number of iterations is approximately the same for both objective functions. Figure 2 explores this behavior in more detail and shows the average number of iterations over 40 topology samples. Again, the number of iterations is approximately the same for both objective functions. Moreover, since the log-log scale is used, Figure 2 indicates the number of iterations increases polynomially with the number of links. Note that Figure 2 only shows the case of $G(\mathbf{f}) = \sum_{\phi\in\Phi}\log(f_\phi)$ for topologies up to 1024 nodes. Due to numerical difficulties, we were not able to solve (2) for 2048 nodes even for a small number of assignments. Thus, we conclude that when $G(\mathbf{f}) = \sum_{\phi\in\Phi}\log(f_\phi)$, the computational bottleneck is not finding new assignments, but solving the basic nonlinear optimization (2).

Note that only one assignment is added at each iteration. Thus, the maximum number of elements in $\mathcal{V}$ is the number of assignments found in Algorithm 2 plus the number of iterations required by Algorithm 1. Hence, we have achieved the goal of determining the solution to (2) for $\mathcal{V} = \bar{\mathcal{V}}$ by computing the solution to (2) for a small set $\mathcal{V}$. Note that the complexity of solving linear and nonlinear optimization problems for a small[5] sets of variables is well known, and hence the complexity of solving (2) is not investigated here.

2) Time to Compute a MWIS: While Algorithm 1 converges after $O(L)$ iterations, each iteration requires solving (14), which has complexity that is no less than the complexity of solving a MWIS problem. As mentioned above, in the worst-case, the MWIS is NP-complete. In order to explore the computational complexity in practical networks, Figure 3 shows the average time to find a new assignment for topologies with between 64 and 2048 nodes and for three types of propagation, namely using a ray-tracing-based urban propagation model, the two-ray propagation model, and the two-ray with lognormal shadowing propagation model. These times were found when solving Algorithm 1 with $G(\mathbf{f}) = \min_{\phi\in\Phi} f_\phi$ and averaged over each topology sample and over each iteration of Algorithm 1. In the case of Figure 3, each point is from averaging over 40 topology samples (840 topologies in total). Figure 3 also shows the 95% confidence intervals.

Figure 3 clearly shows that the time to compute new assignments is quite small, e.g., for a 2048 node network, it takes only one second. Figure 3 also indicates that the computation time grows polynomially with the number of links. While a detailed examination of the typical computation time is beyond the scope of this paper, Figure 3 strongly indicates that finding new assignments is not computationally difficult in practical networks.

It should be noted that the computation times shown in Figure 3 were found with a PC with a two 3.0GHz Intel quadcore processors with 16GB ram. However, each computation only used a single core. Also, CPLEX v10 was used. Other integer linear programming solvers may give dramatically different results.

3) The Number of Multi-Conflicts: As mentioned in Section III-F, in order for the throughput found by solving (2) to match the actual throughput when the schedule is deployed, the assignments used in the schedule must not have any multi-conflicts. The scheme discussed in Section III-F can be used to remove the multi-conflicts. However, each time a multi-conflict is detected and removed, an ILP problem (14) must be solved, increasing the overall computation time. Figure 4 shows the average number of multi-conflicts found (and removed) for various sizes of networks. Roughly, the number of multi-conflicts grows with the number of nodes and the number of gateways. Comparing Figure 4 to Figure 2 we observe that

[5]By small, we mean much less than 2^L.

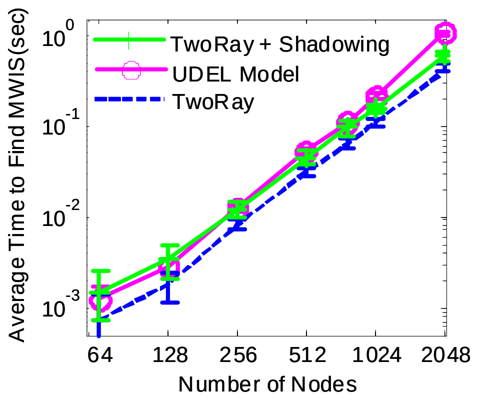

Fig. 3. Computation time to solve the MWIS problem versus the number of nodes in the topology. The vertical bars indicate the 95% confidence intervals.

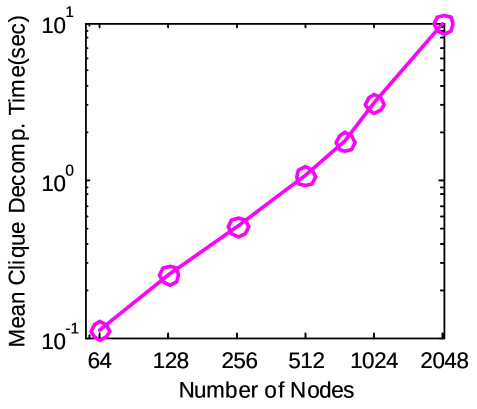

Fig. 5. Time to compute a clique decomposition as a function of the number of nodes in the network.

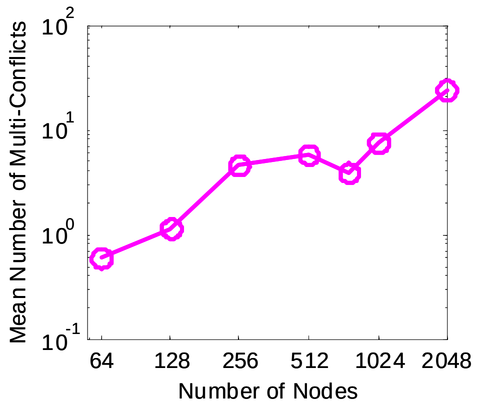

Fig. 4. The average number of multi-conflicts detected and removed for topologies of different sizes.

the number of multi-conflicts is much smaller than the total number of iterations. On the other hand, failing to remove multi-conflicts can severely impact the throughput when the schedule is deployed.

4) Time to Perform Clique Decomposition: As discussed in Section III-E, the time to find a new assignment is greatly reduced if a clique decomposition is performed first. Figure 5 shows that the time required to perform this decomposition is on the order to the time it takes to perform one iteration of Algorithm 1. Since Algorithm 1 requires that tens or hundreds of iterations are performed, the time to compute a single clique decomposition is negligible. However, we do not recompute the clique decomposition every time a multi-conflict is found.

C. Comparison with 802.11 CSMA/CA

With the ability to compute optimal schedules, the impact of optimal schedules on the throughput as compared to 802.11 with CSMA/CA can be investigated. Figure 6 shows the ratio of the optimal throughput to the throughput that 802.11 CSMA/CA can achieve. Here the throughput metric is $\min_{\phi \in \Phi} f_\phi$. Qualnet was used to estimate the throughput of 802.11. RTS/CTS and Qualnet's automatic rate fallback scheme were used[6]. The 802.11 CSMA/CA throughput was determined by sending data to each destination at a constant rate (1000 B packets were used). The sending rate was adjusted until the maximum of $\min_{\phi \in \Phi} f_\phi$ was found. Confidence intervals were generated via bootstrapping [49] to ensure that the estimated throughput was accurate within 10%.

Figure 6 shows that for networks with a large number of gateways, optimal scheduling can have a dramatic improvement in the throughput. On the other hand, the simulations used a standard version of 802.11 CSMA/CA. It is conceivable that if 802.11 is better tuned (e.g., by tuning CCA [50]) and a better version of the ARF is used, then the throughput with 802.11 CSMA/CA could be improved and the relative improvement provided by optimal scheduling would be reduced.

Figure 6 provides a baseline for the range of improvement in throughput that optimal scheduling can achieve. While it is expected that scheduling results in a higher throughput, the degree of the improvement and the dependence on the topology has been unknown. Figure 6 indicates when there are a large number of gateways, scheduling tends to provide tremendous improvements in throughput over 802.11 with CSMA/CA. The scale of this improvement motivates further research on scheduling for networks with many gateways and perhaps supports the extra cost required to deploy hardware capable of performing scheduled transmissions. For example, improvements of this size are large enough that optimal scheduling will likely still provide considerably higher throughput when factors like overhead and errors due to synchronization are accounted for and CSMA/CA is well tuned.

On the other hand, Figure 6 indicates potential difficulties with improving the throughput on networks with few gateways. For example, [6] developed a scheme that achieves at least 1/3 of the optimal throughput (under the condition that co-channel interference does not arise). Figure 6 indicates that such a scheme will only slightly improve the throughput on networks with a small number of gateways. However, improvements of that size might also be possible by tuning CSMA/CA.

V. Conclusions

This paper presented practical techniques for computing optimal schedules in multihop wireless networks even when co-channel interference arises. The algorithms can compute optimal schedules within a few minutes for networks with

[6]In Qualnet v3.95, by default packets larger than 0B use RTS/CTS. Some vendors suggest disabling RTS/CTS.

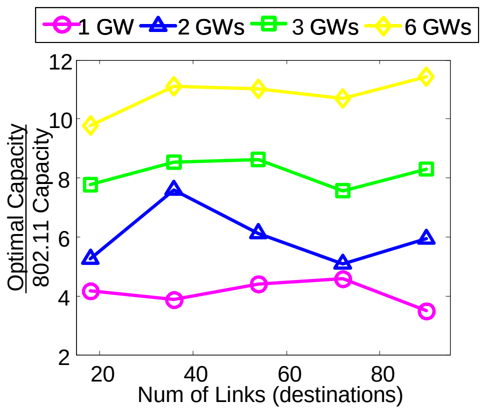

Fig. 6. Comparison between optimal scheduling and 802.11 in mesh networks covering 6×6 block regions of downtown Chicago.

2048 nodes and within a few seconds for networks with 128 nodes.

The performance improvement provided by the optimal scheduling is significant if there are a large number of gateways. For example, as compared to 802.11's CSMA/CA, optimal scheduling improves performance by a factor between 3 and 11, with the improvement increasing as the density of gateways increases.

There are a wide range of computational issues that have yet to be explored. Specific issues to be studied include the impact of the initial set of assignments and techniques to select the modulation scheme. Issues related to the link layer, such as whether and how retransmissions are used, can be found in [15].

VI. Appendix

A. Proof of Theorems 1, 2, 3 and 4

The proofs here are based on the throughput metric $G(\mathbf{f}) = \sum_{\phi\in\Phi} w_\phi \log(f_\phi)$. Thus, to simplify the notation, only a single path routing is considered. The extension to multipath and other throughput metrics is straightforward.

Proof of Theorem 1: The optimal average data rates over each link is a convex sum of the links rates from different assignments, that is, the optimal bit-rate over link x is $\sum_{\mathbf{v}\in\mathcal{V}} \alpha^*_{\mathbf{v}} R(\mathbf{v},x)$, where α^* defines the optimal schedule. In other words, the set of feasible link bit-rates is a convex set where the extreme points are some of the rows of R. Obviously, the vector of optimal link rates is the vector $\sum_{\mathbf{v}\in\mathcal{V}} \alpha^*_{\mathbf{v}} R(\mathbf{v},:) \in \mathbb{R}^L$, the space of vectors with L elements. Due to Caratheodory's Theorem (e.g., Theorem B.6 in [23]), a point within a convex hull in $\mathbb{R}^L$ is specified by at most $L+1$ extreme points. That is, there exists a set, $\mathcal{V}'$ with $L+1$ elements such that

$$\sum_{\mathbf{v}\in\mathcal{V}} \alpha^*_{\mathbf{v}} R(\mathbf{v},:) = \sum_{\mathbf{v}\in\mathcal{V}'} \alpha'_{\mathbf{v}} R(\mathbf{v},:),$$

where $\boldsymbol{\alpha}'$ might be different set of weights from $\boldsymbol{\alpha}^*$. Hence, the optimal link bit-rates found by optimizing over $\overline{\mathcal{V}}$, the set of all possible assignments, can be achieved by only using the set of assignments $\mathcal{V}'$. Thus, the resulting utility is unchanged when $\mathcal{V}'$ is used as opposed to $\overline{\mathcal{V}}$.

Now it is shown that $\mathcal{V}'$ can be selected so that $\mathcal{V}'$ has less than $L+1$ elements. Suppose otherwise, that is, $\mathcal{V}'$ has exactly $L+1$ elements, and $\mathcal{V}'$ is the smallest set such that the optimal schedule is in $Co(\{R(\mathbf{v},:) : \mathbf{v}\in\mathcal{V}'\})$, the convex hull of $\{R(\mathbf{v},:) : \mathbf{v}\in\mathcal{V}'\}$. Since the faces of $Co(\{R(\mathbf{v},:) : \mathbf{v}\in\mathcal{V}'\})$ are defined by no more than L extreme points, the assumption that the optimal bit-rates cannot be specified by L points implies that the optimal bit-rates must be in the interior of $Co(\{R(\mathbf{v},:) : \mathbf{v}\in\mathcal{V}'\})$. That is, there is an open set that contains the optimal point and this open set is in the interior of $Co(\{R(\mathbf{v},:) : \mathbf{v}\in\mathcal{V}'\})$. For example, letting $\mathbf{r}^{**}$ be the vector of optimal bit-rates, the vector $\mathbf{r}^{**}+\varepsilon\mathbf{r}^{**}$ is also in the interior of $Co(\{R(\mathbf{v},:) : \mathbf{v}\in\mathcal{V}'\})$, where $\varepsilon > 0$ is small enough. Since $\mathbf{r}^{**}$ is the optimal vector of bit-rates over the interior of $Co(\{R(\mathbf{v},:) : \mathbf{v}\in\mathcal{V}'\})$, the utility of $\mathbf{r}^{**}$ must be higher than the utility of $\mathbf{r}^{**}+\varepsilon\mathbf{r}^{**}$. However, this is a contradiction since the link bit-rates $\mathbf{r}^{**}+\varepsilon\mathbf{r}^{**}$ result in uniformly large flow rates than $\mathbf{r}^{**}$, which will increase the throughput. Hence, $\mathcal{V}'$ can be selected to have fewer than $L+1$ elements. ■

Proof of Theorem 2 and 3: The relevant Lagrange function is

$$L(\mathbf{f},\alpha,\boldsymbol{\mu},\lambda) = -\sum_{\phi\in\Phi} w_\phi \log(f_\phi) + \lambda\left(\sum_{\mathbf{v}\in\mathcal{V}} \alpha_{\mathbf{v}} - 1\right) + \sum_{x=1}^{L} \mu_x \left(\sum_{\{\phi:x\in P(\phi)\}} f_\phi - \sum_{\mathbf{v}\in\mathcal{V}} \alpha_{\mathbf{v}} R(\mathbf{v},x)\right). \tag{15}$$

After some manipulation, the dual function is found to be

$$q(\boldsymbol{\mu},\lambda) = \inf_{\mathbf{f},\alpha\geq 0} -\sum_{\phi\in\Phi} \log(f_\phi) w_\phi - \lambda + \sum_{x=1}^{L} \mu_x \sum_{\{\phi:x\in P(\phi)\}} f_\phi - \sum_{\mathbf{v}\in\mathcal{V}} \alpha_{\mathbf{v}} \left(\sum_{x=1}^{L} R(\mathbf{v},x)\mu_x - \lambda\right). \tag{16}$$

We immediately note that if $\sum_{x=1}^{L} R(\mathbf{v},x)\mu_x - \lambda > 0$ for some x, then $q(\boldsymbol{\mu},\lambda) = -\infty$. Hence, we restrict the domain of q, to be such that $\sum_{x=1}^{L} R(\mathbf{v},x)\mu_x - \lambda \leq 0$. On the other hand, when solving the dual problem, an objective is to maximize q with respect to λ. It is equivalent to minimizing λ over the domain $\sum_{x=1}^{L} R(\mathbf{v},x)\mu_x - \lambda \leq 0$. Thus,

$$\lambda^* = \max_{\mathbf{v}\in\mathcal{V}} \sum_{x=1}^{L} R(\mathbf{v},x)\mu_x, \tag{17}$$

proving Proposition 2. Furthermore, for this λ, the $\sum_{x=1}^{L} R(\mathbf{v},x)\mu_x - \lambda < 0$ for $\mathbf{v}\notin\mathcal{V}^*$, thus, the infimum in (16) must have $\alpha_{\mathbf{v}} = 0$ for $\mathbf{v}\notin\mathcal{V}^*$, proving Proposition 3. ■

Therefore, we can rewrite the dual function as,

$$q(\mu) = \inf_{\mathbf{f}\geq 0} -\sum_{\phi\in\Phi} \log(f_\phi) w_\phi + \sum_{x=1}^{L} \mu_x \sum_{\{\phi:x\in P(\phi)\}} f_\phi - \max_{\mathbf{v}\in\mathcal{V}} \sum_{x=1}^{L} R(\mathbf{v},x)\mu_x, \tag{18}$$

where $\alpha_{\mathbf{v}}$ has been eliminated since λ^* results in the infimum being achieved for $\alpha_{\mathbf{v}} = 0$.

Proof of Theorem 4: We slightly modify (2) to

$$\begin{aligned}&\min G(\mathbf{f})\\ &\text{subject to:} \sum_{\{\phi: x\in P(\phi)\}} f_\phi - \sum_{\mathbf{v}\in\mathcal{V}} \alpha_{\mathbf{v}} R(\mathbf{v},x) \le \rho_x\\ &\sum_{\mathbf{v}\in\mathcal{V}} \alpha_{\mathbf{v}} - 1 = A,\end{aligned}$$

so (2) is the case where $\boldsymbol{\rho} = 0$ and $A = 0$. We will denote the value of the optimal solution of the above problem as $G^*(\boldsymbol{\rho}, A)$. From sensitivity analysis (e.g., [23]), we have

$$\mu_x^* = \frac{\partial G^*(\rho, A)}{\partial \rho_x} \quad (19)$$

$$\lambda^* = \frac{\partial G^*(\rho, A)}{\partial A}. \quad (20)$$

Equation (19), implies that if the amount of bit-rate that is applied to link x is increased by a small amount ε, then the total utility will increase by $\mu_x^*\varepsilon$. It is critical to note that in this analysis, the bit-rate applied to link x does not come at the expense of bit-rates of other links.

Now consider the multiplier, λ^*. The constraint $\sum_{\mathbf{v}\in\mathcal{V}} \alpha_{\mathbf{v}} = 1 + A$ can be interpreted as the allowing the total bandwidth of size $1 + A$ to be shared among all assignments. Thus, if the bandwidth is increased from size 1 to size $1 + \varepsilon$, then the utility will increase by $\lambda\varepsilon$. Similarly, if the bandwidth is decreased by ε, then the utility will decrease by $\lambda\varepsilon$.

While the analysis above assumed that the extra bandwidth is allocated to link x without impacting the bit-rate of the other links, we now consider the more relevant problem where this extra assignment comes at the expense of other links. Specifically, if we allocate assignment $\mathbf{v}^+$ with ε of the bandwidth, then the total bandwidth allocated to the other assignments must be decreased by ε. In particular, let $\mathcal{V}' = \{\mathbf{v}_1, ...\mathbf{v}_N\}$ and when optimizing over the set of assignments $\mathcal{V}'$, let the associated optimal bandwidth allocated to $\mathbf{v}_i$ be of size α_i^*, where, of course, $\sum_{i=1}^N \alpha_i^* = 1$. Now in order to allocate bandwidth ε to assignment $\mathbf{v}^+$, we adjust the allocation to $\alpha_i^+ = (1-\varepsilon)\alpha_i^*$, and hence the assignments $\{\mathbf{v}^+, \mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_N\}$ are allocated bandwidths of width $\{\varepsilon, (1-\varepsilon)\alpha_1^*, (1-\varepsilon)\alpha_2^*, ..., (1-\varepsilon)\alpha_N^*\}$, respectively. Based on the discussion above, the change in utility is

$$\varepsilon\left(\sum_{x=1}^{L} R\left(\mathbf{v}^+, x\right)\mu_x^* - \lambda\right), \quad (21)$$

which is positive if (6) holds. ■

B. Proof of Theorem 6

Consider the problem

$$\begin{aligned}&\max g(\boldsymbol{\mu}, \lambda) \quad (22)\\ &\text{subject to } \sum_{x=1}^{L} R(\mathbf{v}, x)\mu_x \le \lambda \text{ for all } \mathbf{v}\in\mathcal{V}\\ &h(\boldsymbol{\mu}) = 0\end{aligned}$$

and Algorithm 3 for solving this problem with $\mathcal{V} = \overline{\mathcal{V}}$.

This problem is the dual of (2) for either objective function by correctly defining g and h. Specifically, if

Algorithm 3

```
1: Select an initial set of assignments $\mathcal{V}(0)$, set $k = 0$.
2: Find $\boldsymbol{\mu}(k)$ and $\lambda(k)$, the solutions to (22) for $\mathcal{V} = \mathcal{V}(k)$.
3: Find $\mathbf{v}(k) = \arg\max_{\mathbf{v}\in\overline{\mathcal{V}}} \sum_{x=1}^{L} \mu_x(k) R(\mathbf{v}, x) - \lambda(k)$.
4: if $\sum_{x=1}^{L} \mu_x(k) R(\mathbf{v}(k)) - \lambda(k) \le 0$, then
     Stop
5: else
     Set $\mathcal{V}(k+1) = \mathcal{V}(k+1)) \cup \mathbf{v}^+(k)$ Go to Step 2
6: end if
```

$g(\boldsymbol{\mu}, \lambda) = \lambda$ and $h(\boldsymbol{\mu}) = \sum_x \mu_x\beta_x - 1$, then (22) is the dual of (2) when $G(\mathbf{f}) = \min_{\phi\in\Phi} f_\phi$. On the other hand, if $g(\boldsymbol{\mu}, \lambda) = -\sum_{\phi\in\Phi} \log\left(\frac{1}{\sum_{x\in P(\phi)} \mu_x}\right) + \sum_{x=1}^{L} \mu_x \sum_{\{\phi: x\in P(\phi)\}} \frac{1}{\sum_{y\in P(\phi)} \mu_y} - \lambda$ and $h(\boldsymbol{\mu}) \equiv 0$, then (22) is the dual of (2) with $G(f) = \sum_{\phi\in\Phi} \log(f_\phi)$. Thus the following theorem applies to both cases.

Theorem 11: The sequence $\{(\boldsymbol{\mu}(n), \lambda(n)) | n = 0, 1, ...\}$ given by Algorithm 3 converges to the optimal solution. Thus, Algorithm 1 converges to the optimal solution.

Lemma 12: Assume that $G(\mathbf{f}) = \sum_{\phi\in\Phi} \log(f_\phi)$. Then the set $\{(\boldsymbol{\mu}(n), \lambda(n)) | n = 1, 2, ...\}$ is a bounded set.

Proof: We first get a lower bound on the optimal value of $G(\mathbf{f})$. Let $\mathbf{v}_x$ be the assignment where link x is active individually. Set $\underline{f} = \min_\phi \min_{x\in P(\phi)} \frac{R(\mathbf{v}_x, x)}{\beta_x}$. Then $\underline{G} = \sum_{\phi\in\Phi} \log\left(\underline{f}\right)$ is a lower bound on $G(\mathbf{f}(\infty))$. Let r^* be the maximum data rate over any link. Then $\overline{G} = \sum_{\phi\in\Phi} \log(r^*)$ is an upper bound on $G(\mathbf{f}(\infty))$. Thus, for a flow θ, we must have that $\underline{G} < \log(f_\theta^*) + \sum_{\phi\in\Phi\backslash\theta} \log(r^*)$. Hence $f_\theta^* \ge \exp\left(\underline{G} - \sum_{\phi\in\Phi\backslash\theta} \log(r^*)\right)$. Since $f_\theta^* = 1/\sum_{x\in P(\theta)} \mu_x^*$ and $\mu_x^* \ge 0$, we have $\mu_x^* < 1/\exp\left(\underline{G} - \sum_{\phi\in\Phi\backslash\theta} \log(r^*)\right)$. Moreover, since $\sum_{x=1}^{L} R(\mathbf{v}, x)\mu_x^* = \lambda^*$, we must have that $\lambda^* \le Lr^*/\exp\left(\underline{G} - \sum_{\phi\in\Phi\backslash\theta} \log(r^*)\right)$. ■

Lemma 13: Assume that $G(f) = \min_{\phi\Phi}(f_\phi)$. Then the set $\{(\boldsymbol{\mu}(n), \lambda(n)) | n = 1, 2, ...\}$ is a bounded set.

Proof: Let $\sum_{x=1}^{L} \mu_x^*\beta_x = F^* \le r^*$ where r^* is the maximum data rate over any link. Thus, $\mu_x \le r^*/\beta_x$.Also, $\lambda^* \le r^*$. ■

Proof of Theorem 11: Since $\{(\boldsymbol{\mu}(n), \lambda(n)) : n = 1, 2, ...\}$ is a bounded sequence, there exists a convergent subsequence. Thus, assume that $\{(\boldsymbol{\mu}(n_j), \lambda(n_j)) : j = 1, 2, ...\}$ is such a sequence and $\lim_{j\to\infty} (\boldsymbol{\mu}(n_j), \lambda(n_j)) = (\boldsymbol{\mu}', \lambda')$. Define a sequence of sets of assignments $\mathcal{V}(n_j) := \mathcal{V}(0) \cup \bigcup_{i=1}^{n_j} \mathbf{v}(i)$. We will show that $\sum_{x=1}^{L} R(\mathbf{v}, x)\mu_x' < \lambda'$ for $\mathbf{v}\in\mathcal{V}(n)$. To this end define $u(\boldsymbol{\mu}, \lambda) := \max_{\mathbf{v}\in\overline{\mathcal{V}}} \sum_{x=1}^{L} R(\mathbf{v}, x)\mu_x - \lambda$. It is straightforward to check that u is a continuous function. Also, $u(\boldsymbol{\mu}(n_j), \lambda(n_j)) = \sum_{x=1}^{L} R(\mathbf{v}(n_j), x)\mu_x(n_j) - \lambda(n_j)$, and since $\mathcal{V}(n_j)$ is increasing in j, $\sum_{x=1}^{L} R(\mathbf{v}, x)\mu_x' - \lambda' \le 0$ for all $\mathbf{v}\in\mathcal{V}(n_j)$ for all j. Therefore, the following string holds

$$\begin{aligned}
& u\left(\left(\boldsymbol{\mu}', \lambda'\right)\right) \\
= \; & u\left(\left(\boldsymbol{\mu}\left(n_j\right), \lambda\left(n_j\right)\right)\right) + u\left(\left(\boldsymbol{\mu}', \lambda'\right)\right) \\
& -u\left(\left(\boldsymbol{\mu}\left(n_j\right), \lambda\left(n_j\right)\right)\right) \\
= \; & \sum_{x=1}^{L} R\left(\mathbf{v}\left(n_j\right), x\right) \mu_x\left(n_j\right) - \lambda\left(n_j\right) \\
& + \left(u\left(\left(\boldsymbol{\mu}', \lambda'\right)\right) - u\left(\left(\boldsymbol{\mu}\left(n_j\right), \lambda\left(n_j\right)\right)\right)\right) \\
\leq \; & \left(\sum_{x=1}^{L} R\left(\mathbf{v}\left(n_j\right), x\right) \mu_x\left(n_j\right) - \lambda\left(n_j\right)\right) \\
& - \left(\sum_{x=1}^{L} R\left(\mathbf{v}\left(n_j\right), x\right) \mu'_x - \lambda'\right) \\
& + \left(u\left(\left(\boldsymbol{\mu}', \lambda'\right)\right) - u\left(\left(\boldsymbol{\mu}\left(n_j\right), \lambda\left(n_j\right)\right)\right)\right) \\
= \; & \left(\sum_{x=1}^{L} R\left(\mathbf{v}\left(n_j\right), x\right)\left(\mu_x\left(n_j\right) - \mu'_x\right) - \left(\lambda\left(n_j\right) - \lambda'\right)\right) \\
& + \left(u\left(\left(\boldsymbol{\mu}', \lambda'\right)\right) - u\left(\left(\boldsymbol{\mu}\left(n_j\right), \lambda\left(n_j\right)\right)\right)\right).
\end{aligned}$$

Since the entries of R are bounded and since u is continuous, the right-hand side converges to zero as $j \to \infty$. Therefore, $u\left(\left(\boldsymbol{\mu}', \lambda'\right)\right) \leq 0$.

Note that $g\left(\left(\boldsymbol{\mu}\left(n_j\right), \lambda\left(n_j\right)\right)\right)$ is a nondecreasing function (since more constraints are added at each iteration) and $g\left(\left(\boldsymbol{\mu}\left(n_j\right), \lambda\left(n_j\right)\right)\right) \leq g\left(\left(\boldsymbol{\mu}\left(\infty\right), \lambda\left(\infty\right)\right)\right)$, where $\left(\boldsymbol{\mu}\left(\infty\right), \lambda\left(\infty\right)\right)$ is the solution to (22) for $\mathcal{V} = \overline{\mathcal{V}}$. Since g is continuous, $\lim_{j\to\infty} g\left(\left(\boldsymbol{\mu}\left(n_j\right), \lambda\left(n_j\right)\right)\right) = g\left(\left(\boldsymbol{\mu}', \lambda'\right)\right) \leq g\left(\left(\boldsymbol{\mu}\left(\infty\right), \lambda\left(\infty\right)\right)\right)$. Thus $\left(\boldsymbol{\mu}', \lambda'\right)$ solves (22) for $\mathcal{V} = \overline{\mathcal{V}}$. And hence, $\left(\boldsymbol{\mu}', \lambda'\right)$ solves the dual of (22), which has a unique solution, hence $\left(\boldsymbol{\mu}\left(\infty\right), \lambda\left(\infty\right)\right) = \left(\boldsymbol{\mu}', \lambda'\right)$. Thus, all subsequences of $\left\{\left(\boldsymbol{\mu}\left(n\right), \lambda\left(n\right)\right) : j = 1, 2, ...\right\}$ converge to $\left(\boldsymbol{\mu}', \lambda'\right)$. Hence, $\lim_{n\to\infty}\left(\boldsymbol{\mu}\left(n\right), \lambda\left(n\right)\right) = \left(\boldsymbol{\mu}', \lambda'\right)$. It is straightforward to show that if $\left\{\left(\mathbf{f}\left(n\right), \boldsymbol{\alpha}\left(n\right)\right) : n = 1, 2, ...\right\}$ is the sequence of solutions to the primal problems via Algorithm 1, then $\lim_{n\to\infty}\left(\mathbf{f}\left(n\right), \boldsymbol{\alpha}\left(n\right)\right) = \left(\mathbf{f}\left(\infty\right), \boldsymbol{\alpha}\left(\infty\right)\right)$, which is the optimal solution to the primal problem. ■

C. Proof of Theorems 7 and 9

Define $\Delta(n) = \arg\max_{\mathbf{v}\in\bar{\mathcal{V}}} R\left(\mathbf{v}, :\right) \boldsymbol{\mu}\left(n\right) - \lambda(n)$. Define $f(n)$ to be the vector of flow rates found during the nth iteration of Algorithm 1 and let $f(\infty) := \lim_{n\to\infty} f(n)$. Thus $G\left(\mathbf{f}\left(n\right)\right)$ to be the optimal value of (2) after the nth iteration. And let $G\left(\mathbf{f}\left(\infty\right)\right)$ be the solution to the full problem. Define $\Delta\lambda_n = \lambda\left(n+1\right) - \lambda\left(n\right)$ and $\boldsymbol{\Delta\mu}\left(n\right) = \boldsymbol{\mu}\left(n+1\right) - \boldsymbol{\mu}\left(n\right)$. Let $\mathbf{v}\left(n\right) = \arg\max_{\mathbf{v}\in\bar{\mathcal{V}}} R\left(\mathbf{v}, :\right) \boldsymbol{\mu}\left(n\right)$. That is, $\mathbf{v}\left(n\right)$ is the assignment added at the nth iteration.

Theorem 14: Let $G\left(\mathbf{f}\right) = \sum_{\phi\in\Phi} \log\left(f_\phi\right)$, then for $\mathbf{f}\left(n\right)$ found by Algorithm 1,$G\left(\mathbf{f}\left(\infty\right)\right) - G\left(\mathbf{f}\left(n\right)\right) \leq \Delta\left(n\right)$.

Proof: The dual of (2) is

$$\max \min_{\mathbf{f}} - \sum_{\phi} \log\left(f_\phi\right) + \sum_{x} \mu_x \sum_{\left\{\phi : x\in P\left(\phi\right)\right\}} f_\phi - \lambda$$

$$\text{subject to } \sum_{x} R\left(\mathbf{v}, x\right) \mu_x \leq \lambda \text{ for all } \mathbf{v} \in \mathcal{V}.$$

Then as above, $\left(\left(\boldsymbol{\mu}\left(n\right), \max_{\mathbf{v}\in\bar{\mathcal{V}}} R\left(\mathbf{v}, :\right) \boldsymbol{\mu}\left(n\right)\right)\right)$ is a feasible (but not optimal) solution to the dual of the full problem. Nonetheless,

$$G\left(\mathbf{f}\left(\infty\right)\right) - G\left(\mathbf{f}\left(n\right)\right) \leq \max_{\mathbf{v}\in\bar{\mathcal{V}}} R\left(\mathbf{v}, :\right) \boldsymbol{\mu}\left(n\right) - \lambda\left(n\right).$$

■

To proof the above for the case when $G\left(\mathbf{f}\right) = \min_{\phi\in\Phi} f_\phi$ we rewrite (2) to

$$\begin{aligned}
& \min -F \\
\text{subject to:} \quad & \sum_{\mathbf{v}\in\mathcal{V}} \alpha_{\mathbf{v}} R\left(\mathbf{v}, x\right) \geq \beta_x F \\
& \sum_{\mathbf{v}\in\mathcal{V}} \alpha_{\mathbf{v}} \leq 1
\end{aligned} \tag{23}$$

where β_x is the number of flows that pass through link x divided by the bit-rate of link x. Thus, with normalization, $R\left(\mathbf{v}, x\right) \in \{0, 1\}$.

Theorem 15: Let $G\left(\mathbf{f}\right) = \min_{\phi\in\Phi} f_\phi$, then for $\mathbf{f}\left(n\right)$ found by Algorithm 1,$G\left(\mathbf{f}\left(\infty\right)\right) - G\left(\mathbf{f}\left(n\right)\right) \leq \Delta\left(n\right)$

Proof: The dual of (2) with $\mathcal{V} = \bar{\mathcal{V}}$ is

$$\begin{aligned}
& & \min \lambda \\
-R\left(\mathbf{v}, :\right) \boldsymbol{\mu} & \geq & -\lambda \text{ for all } \mathbf{v} \in \bar{\mathcal{V}} \\
\sum_{x} \mu_x \beta_x & \geq & 1.
\end{aligned}$$

Note that $\left(\boldsymbol{\mu}\left(n\right), \max_{\mathbf{v}\in\bar{\mathcal{V}}} R\left(\mathbf{v}, :\right) \boldsymbol{\mu}\left(n\right)\right)$ is a feasible (but not optimal) solution to the dual of the full problem. Thus, $G\left(\mathbf{f}\left(\infty\right)\right) \leq \max_{\mathbf{v}\in\bar{\mathcal{V}}} R\left(\mathbf{v}, :\right) \boldsymbol{\mu}\left(n\right)$ and $G\left(\mathbf{f}\left(\infty\right)\right) - G\left(\mathbf{f}\left(n\right)\right) \leq \max_{\mathbf{v}\in\bar{\mathcal{V}}} R\left(\mathbf{v}, :\right) \boldsymbol{\mu}\left(n\right) - G\left(\mathbf{f}\left(n\right)\right) = \Delta\left(n\right)$. ■

Theorem 9 follows from Theorems 14 and 15. We now focus on proving Theorem 7. To this end, the following lemmas are proved.

Lemma 16: $\Delta\lambda\left(n\right) \geq R\left(\mathbf{v}, :\right) \boldsymbol{\Delta\mu}\left(n\right)$ for all $\mathbf{v} \neq \mathbf{v}\left(n\right)$ and $\mathbf{v} \in \mathcal{V}^*\left(n+1\right)$

Proof: From the identity, $\lambda\left(n\right) = \max_{\mathbf{v}\in\mathcal{V}(n)} R\left(\mathbf{v}, :\right) \boldsymbol{\mu}\left(n\right) = R\left(\mathbf{v}, :\right) \boldsymbol{\mu}\left(n\right)$ for $\mathbf{v} \in \mathcal{V}^*\left(n\right)$, we have for $\mathbf{v} \in \mathcal{V}^*\left(n+1\right)$ and $\mathbf{v} \neq \mathbf{v}\left(n\right)$

$$\begin{aligned}
& \lambda\left(n+1\right) - \lambda\left(n\right) \\
= \; & R\left(\mathbf{v}, :\right) \boldsymbol{\mu}\left(n+1\right) - \max_{\mathbf{v}\in\mathcal{V}(n)} R\left(\mathbf{v}, :\right) \boldsymbol{\mu}\left(n\right) \\
\geq \; & R\left(\mathbf{v}, :\right) \boldsymbol{\mu}\left(n+1\right) - R\left(\mathbf{v}, :\right) \boldsymbol{\mu}\left(n\right).
\end{aligned}$$

■

Lemma 17: $\Delta\lambda\left(n\right) = R\left(\mathbf{v}\left(n\right), :\right) \boldsymbol{\Delta\mu}\left(n\right) + \Delta\left(n\right)$

Proof: The newly added assignment, $\mathbf{v}\left(n\right)$, is always an active assignment in the schedule found in the$\left(n+1\right)$th iteration, i.e., $\mathbf{v}\left(n\right) \in \mathcal{V}^*\left(n+1\right)$. Thus, $\lambda\left(n+1\right) = R\left(\mathbf{v}\left(n\right), :\right) \boldsymbol{\mu}\left(n+1\right)$. From the definition of $\Delta\left(n\right)$, we have $-\lambda\left(n\right) = -R\left(\mathbf{v}\left(n\right), :\right) \boldsymbol{\mu}\left(n\right) + \Delta\left(n\right)$. Thus, $\lambda\left(n+1\right) - \lambda\left(n\right) = R\left(\mathbf{v}\left(n\right), :\right) \boldsymbol{\mu}\left(n+1\right) - R\left(\mathbf{v}\left(n\right), :\right) \boldsymbol{\mu}\left(n\right) + \Delta\left(n\right)$. ■

Lemma 18: $\boldsymbol{\beta}^T \boldsymbol{\Delta\mu}\left(n\right) = 0$.

Proof: The Lagrangian for (23) is

$$\begin{aligned}
L\left(\boldsymbol{\mu}, \lambda, \boldsymbol{\alpha}, F\right) & = & -F + \sum_{x} \mu_x \left(\beta_x F - \sum_{\mathbf{v}} \alpha_{\mathbf{v}} R\left(\mathbf{v}, x\right)\right) \\
& & +\lambda \left(\sum_{\mathbf{v}} \alpha_{\mathbf{v}} - 1\right).
\end{aligned}$$

Since F appears linearly, for the optimal value of $\boldsymbol{\mu}$ we must have that $-1+\sum_x \beta_x \mu_x = 0$. Thus, for all n, $\boldsymbol{\beta}^T \boldsymbol{\mu}(n) = 1$, and hence $\boldsymbol{\beta}^T \Delta\boldsymbol{\mu}(\ n) = 0$. ■

Note that the set of active assignments $\mathcal{V}^*(n+1)$ and the corresponding matrix of data-rates must be schedulable in the sense that sufficient data must flow on each link. This gives rise to following condition on $\mathcal{V}^*(n+1)$ and R.

Condition 19: There exists a vector $\boldsymbol{\alpha}$ with $\sum_{\mathbf{v}\in\mathcal{V}^*(n+1)} \alpha_{\mathbf{v}} = 1$ and $\alpha_{\mathbf{v}} > 0$ for all $\mathbf{v}\in\mathcal{V}^*(n+1)$ such that there exists a $t>0$ such that

$$\boldsymbol{\alpha} R(:,x) \geq t\beta_x \text{ for all } x. \tag{24}$$

Lemma 20: There exists a $q>0$ that is independent of $\mathcal{V}^*(n+1)$ such that $\max_{\mathbf{v}\in\mathcal{V}^*(n+1)\backslash\mathbf{v}(n)} R(\mathbf{v},:)\Delta\boldsymbol{\mu}(n) \geq -qR(\mathbf{v}(n),:)\Delta\boldsymbol{\mu}(n)$.

Proof: Multiplying both sides of (24) by $\Delta\mu_x(n)$ and summing results in

$$\sum_{x=1}^{L}\Delta\mu_x(n)\sum_{\mathbf{v}\in\mathcal{V}^*(n+1)}\alpha_{\mathbf{v}} R(\mathbf{v},x) \geq t\sum_{x=1}^{L}\beta_x\Delta\mu_x(n)$$

and from Lemma 18, we have

$$\sum_{\mathbf{v}\in\mathcal{V}^*(n+1)\backslash\mathbf{v}(n)}\alpha_{\mathbf{v}}\sum_x\Delta\mu_x(n)R(\mathbf{v},x) + \alpha_{\mathbf{v}(n)}\Delta\mu_x(n)R(\mathbf{v}(n),x) \geq 0.$$

Thus,

$$\max_{\mathbf{v}\in\mathcal{V}^*(n+1)\backslash\mathbf{v}(n)} R(\mathbf{v},:)\Delta\boldsymbol{\mu}(n) \geq -q(R,\boldsymbol{\beta})R(\mathbf{v}(n),x)\Delta\mu_x(n)$$

where $q(R,\boldsymbol{\beta})>0$ is a constant that depends on the vector $\boldsymbol{\alpha}$ given by Condition 19, and hence depends on the matrix of active assignments $\mathcal{V}^*(n+1)$ and the vector $\boldsymbol{\beta}$. The set of active assignments is in the set $\mathrm{V}(\boldsymbol{\beta},L)$ where

$$\mathcal{V}(\boldsymbol{\beta},L) := \left\{\{0,1\}^{s\times L} \mid s\leq L, \text{ Condition 19 holds}\right\},$$

where s is the number of active assignments, and $s\leq L$. Clearly, $\mathrm{V}(\boldsymbol{\beta},L)$ is a compact set (actually, it is a finite set). Hence, there exists a $q := \min_{R\in\mathcal{V}(\boldsymbol{\beta},L)} q(R,\boldsymbol{\beta})$ where $q>0$ and thus $\max_{\mathbf{v}\in\mathcal{V}^*(n+1)\backslash\mathbf{v}(n)} R(\mathbf{v},:)\boldsymbol{\Delta\mu}(n) \geq -q\Delta\mu_x(n)R(\mathbf{v}(n),x)$ for any $\mathcal{V}^*(n+1) \in\mathrm{V}(\boldsymbol{\beta},L)$. ■

Lemma 21: $\Delta\lambda(n)\geq\delta\Delta(n)$ for some $\delta>0$.

Proof: From Lemmas 16, 17, and 20

$$\begin{aligned}\Delta\lambda(n) &\geq \max_{\mathbf{v}\in\mathcal{V}^*(n+1)\backslash\mathbf{v}(n)} R(\mathbf{v},:)\Delta\boldsymbol{\mu}(n)\\ &\geq -qR(\mathbf{v}(n),:)\Delta\boldsymbol{\mu}(n)\\ &= q(\Delta(n)-\Delta\lambda(n))\\ \Delta\lambda(n)(1+q) &\geq q\Delta(n)\\ \Delta\lambda(n) &\geq \frac{q}{q+1}\Delta(n).\end{aligned}$$

■

Proof of Theorem 7: Since there is no duality gap, $\lambda(n) = G(\mathbf{f}(n))$. Thus, from Lemma 21

$$\frac{G(\mathbf{f}(n+1))-G(\mathbf{f}(n))}{\Delta(n)} \geq \delta.$$

On the other hand, by Theorem 15, $G(\mathbf{f}(\infty)) - G(\mathbf{f}(n)) \leq \Delta(n)$. Therefore, we have

$$\begin{aligned}&\frac{G(\mathbf{f}(n+1))-G(\mathbf{f}(n))}{G(\mathbf{f}(\infty))-G(\mathbf{f}(n))} \geq \delta\\ &\frac{(G(\mathbf{f}(\infty))-G(\mathbf{f}(n)))-(G(\mathbf{f}(\infty))-G(\mathbf{f}(n+1)))}{G(\mathbf{f}(\infty))-G(\mathbf{f}(n))} \geq \delta\\ &1-\frac{(G(\mathbf{f}(\infty))-G(\mathbf{f}(n+1)))}{G(\mathbf{f}(\infty))-G(\mathbf{f}(n))} \geq \delta\\ &\frac{(G(\mathbf{f}(\infty))-G(\mathbf{f}(n+1)))}{G(\mathbf{f}(\infty))-G(\mathbf{f}(n))} \leq 1-\delta < 1.\end{aligned}$$

■

Note that in practice, numerical errors limit the accuracy of the solutions. These errors result in $G(\mathbf{f}(\infty)) - G(\mathbf{f}(n+1))$ approaching zero slowly for large n. Thus, if $\Delta_{Threshold}$ is very small, it may take many iterations before $\Delta(n) < \Delta_{Threshold}$. Thus, $\Delta_{Threshold}$ should not be to small. While further research is required to understand the source and impact of numerical errors, we suspect that errors in channel gain measurements and node synchronization result in more significant reduction in actual throughput than using a large value of $\Delta_{Threshold}$.

D. Proof of Theorem 8

Lemma 22: $\|\boldsymbol{\mu}(n+1)-\boldsymbol{\mu}(n)\| \geq \delta|\lambda(n+1)-\lambda(n)|$ for some $\delta>0$.

Proof: Recall that $\lambda(n) = \max_{\mathbf{v}\in\mathcal{V}(n)} R(\mathbf{v},:)\boldsymbol{\mu}(n)$ and for $\mathbf{v}\in\mathcal{V}^*(n)$ we have $\lambda(n) = R(\mathbf{v},:)\boldsymbol{\mu}(n)$. Let $\mathbf{v}'\in\mathcal{V}^*(n+1)\cap\mathcal{V}(n)$. Then

$$R(\mathbf{v}',:)\boldsymbol{\mu}(n+1) - R(\mathbf{v}',:)\boldsymbol{\mu}(n) \geq \lambda(n+1)-\lambda(n).$$

Since $R(\mathbf{v}',x)\in\{0,1\}$, there exists an x such that $\mu_x(n+1)-\mu_x(n) \geq \frac{1}{L}(\lambda(n+1)-\lambda(n))$, and $\|\boldsymbol{\mu}(n+1)-\boldsymbol{\mu}(n)\| \geq \frac{1}{L}|(\lambda(n+1)-\lambda(n))|$. ■

Lemma 23: Let $A\in\{0,1\}^{|\Phi|\times L}$ with $A(\phi,x)=1$ if $x\in P(\phi)$ and $A(\phi,x)=0$ otherwise. Suppose that the null space of A is empty. Then $A(\phi,:)\boldsymbol{\mu} = 1/f_\phi$ and there exists a $\delta>0$ such that $\|\mathbf{f}(n)-\mathbf{f}(n+1)\| \geq \delta\|\boldsymbol{\mu}(n)-\boldsymbol{\mu}(n+1)\|$.

Proof: Since the null space of A is empty, all the singular values of A are nonzero. Thus, $\left(\sum_{\phi\in\Phi}\left(\frac{1}{f_\phi(k)}-\frac{1}{f_\phi(k+1)}\right)^2\right)^{1/2} \geq \frac{1}{\underline{\sigma}}\|\boldsymbol{\mu}(k)-\boldsymbol{\mu}(k+1)\|$, where $\underline{\sigma}$ is the smallest singular value of A.

Recall that the proof of Lemma 12 showed that there exists a $\underline{f}$ such that $f_\phi(n)\geq\underline{f}$ for all n and ϕ. Direct calculation shows that $|f_\phi(n)-f_\phi(n+1)| \geq \underline{f}^2\left|\frac{1}{f_\phi(n)}-\frac{1}{f_\phi(n+1)}\right|$. Thus, $\|\mathbf{f}(n)-\mathbf{f}(n+1)\| \geq \frac{\underline{f}^2}{\underline{\sigma}}\|\boldsymbol{\mu}(k)-\boldsymbol{\mu}(k+1)\|$. ■

Combining the previous lemmas we get the following.

Lemma 24: If the assumption of Lemma 23 holds, then $\|\mathbf{f}(n)-\mathbf{f}(n+1)\| \geq \delta\|(\boldsymbol{\mu}(n+1),\lambda(n+1))-(\boldsymbol{\mu}(n),\lambda(n))\|$ for some $\delta\geq 0$.

Proof of Theorem 8: [7]Define $u(\boldsymbol{\mu},\lambda) := \max_{\mathbf{v}\in\overline{\mathcal{V}}}\sum_{x=1}^{L}R(\mathbf{v},x)\mu_x - \lambda$. Let $(\boldsymbol{\mu}(0),\lambda(0))$ be the multipliers that result from solving (2)

[7]This proof is based on a proof in [51].

with $\mathcal{V} = \{\mathbf{v} : v_x = 1 \text{ for exact one } x\}$. Let $\lambda_o = \max_{\mathbf{v}\in\mathcal{V}} \sum_{x=1}^{L} R(\mathbf{v},x)\,\mu_x(0)$. Then, $u((\boldsymbol{\mu}(0), 2\lambda_o)) = -\lambda_o$. Thus, for each n, there exists a $\gamma(n) \in [0,1]$ such that

$$0 = u(\gamma(n)(\boldsymbol{\mu}(n), \lambda(n)) + (1-\gamma(n))(\boldsymbol{\mu}(0), 2\lambda_o)). \tag{25}$$

From (25),

$$\begin{aligned} 0 &= u(\gamma(n)(\boldsymbol{\mu}(n), \lambda(n)) + (1-\gamma(n))(\boldsymbol{\mu}(0), 2\lambda_o)) \\ &\le \gamma(n)\,u(\boldsymbol{\mu}(n), \lambda(n)) + (1-\gamma(n))\,u(\boldsymbol{\mu}(0), 2\lambda_o), \end{aligned}$$

where the inequality is implied by the convexity of u. Therefore,

$$u(\boldsymbol{\mu}(n), \lambda(n)) \ge -\frac{(1-\gamma(n))}{\gamma(n)} u(\boldsymbol{\mu}(0), 2\lambda_o).$$

Since $\mathbf{v}(n) \in \mathcal{V}^*(n+1)$, we have $R(\mathbf{v}(n), :)\,\boldsymbol{\mu}(n+1) - \lambda(n+1) = 0$. Also, $u((\boldsymbol{\mu}(n), \lambda(n))) = R(\mathbf{v}(n), :)\,\boldsymbol{\mu}(n+1) - \lambda(n+1)$. Therefore,

$$\begin{aligned} & -\frac{(1-\gamma(n))}{\gamma(n)} u(\boldsymbol{\mu}(0), 2\lambda_o) \\ \le\ & u(\boldsymbol{\mu}(n), \lambda(n)) \\ =\ & R(\mathbf{v}(n), :)\,\boldsymbol{\mu}(n) - \lambda(n) \\ & - (R(\mathbf{v}(n), :)\,\boldsymbol{\mu}(n+1) - \lambda(n+1)) \\ =\ & R(\mathbf{v}(n), :)(\boldsymbol{\mu}(n) - \boldsymbol{\mu}(n+1)) - (\lambda(n) - \lambda(n+1)) \\ \le\ & r^* \left\|(\boldsymbol{\mu}(n+1), \lambda(n+1)) - (\boldsymbol{\mu}(n), \lambda(n))\right\|, \end{aligned}$$

where r^* is the highest data rate across any link, and hence $R(\mathbf{v}, x) \le r^*$. From the above and Lemma 24 we have

$$\begin{aligned} \|\mathbf{f}(n) - \mathbf{f}(n+1)\| &\ge -\frac{\delta(1-\gamma(n))}{\gamma(n)} u(\boldsymbol{\mu}(0), 2\lambda_o) \\ &\ge -\delta(1-\gamma(n))\,u(\boldsymbol{\mu}(0), 2\lambda_o) \end{aligned}$$

for some $\delta > 0$. Or

$$(1-\gamma(n)) \le \frac{\|\mathbf{f}(n) - \mathbf{f}(n+1)\|}{-\delta u(\boldsymbol{\mu}(0), 2\lambda_o)}. \tag{26}$$

Corresponding to the point $(\boldsymbol{\mu}(0), 2\lambda_o)$, define flow rates $\tilde{\mathbf{f}}$ where $\tilde{f}_\phi = \frac{1}{\sum_{x\in P(\phi)} \mu_x(0)}$. Clearly, this set of data rates is suboptimal but feasible. Similarly, $\mathbf{f}(n)$ is suboptimal but feasible. Hence, $\gamma(n)\mathbf{f}(n) + (1-\gamma(n))\tilde{\mathbf{f}}$ is suboptimal but feasible. Therefore,

$$-G(\mathbf{f}(\infty)) \le -G\left(\gamma(n)\mathbf{f}(n) + (1-\gamma(n))\tilde{\mathbf{f}}\right),$$

where $\mathbf{f}(\infty)$ is the vector of optimal flow rates. Then

$$\begin{aligned} & (-G(\mathbf{f}(\infty))) - (-G(\mathbf{f}(n))) \qquad (27) \\ \le\ & \left(-G\left(\gamma(n)\mathbf{f}(n) + (1-\gamma(n))\tilde{\mathbf{f}}\right)\right) - (-G(\mathbf{f}(n))) \\ \le\ & K\left\|\gamma(n)\mathbf{f}(n) + (1-\gamma(n))\tilde{\mathbf{f}} - \mathbf{f}(n)\right\| \\ =\ & K(1-\gamma(n))\left\|\mathbf{f}(n) - \tilde{\mathbf{f}}\right\|, \end{aligned}$$

where $K = \max_{\mathbf{f}\in\{\mathbf{f}|\underline{f}\le f_\phi\le r^*\}} \|\nabla G(\mathbf{f})\|$ and $\nabla G(\mathbf{f})$ is the gradient of G at $\mathbf{f}$ and $\underline{f}$ is the lower bound on the flow rates given in Lemma 12.

Combining (26) and (27) yields,

$$\begin{aligned} & (-G(\mathbf{f}(\infty))) - (-G(\mathbf{f}(n))) \qquad (28) \\ \le\ & \frac{K}{-\delta u(\boldsymbol{\mu}(0), 2\lambda_o)} \|\mathbf{f}(n) - \mathbf{f}(n+1)\| \left\|\mathbf{f}(n) - \tilde{\mathbf{f}}\right\| \end{aligned}$$

Define $D(n) := (-G(\mathbf{f}(\infty))) - (-G(\mathbf{f}(n)))$. Thus, (28) implies

$$D(n) \le K_1 \|\mathbf{f}(n) - \mathbf{f}(n+1)\|, \tag{29}$$

where $K_1 = \frac{K}{-\delta u(\boldsymbol{\mu}(0), 2\lambda_o)} \max_{\{\mathbf{f}|\underline{f}\le f_\phi\le r^*\}} \left\|\mathbf{f} - \tilde{\mathbf{f}}\right\|$.

On the other hand, over the domain $\left\{\mathbf{f}\,\middle|\,\underline{f} \le f_\phi \le r^*\right\}$, $\sum_{\phi\in\Phi} \log(f_\phi)$ is a strongly convex function (see Proposition B.5 in [23]). Thus,

$$\begin{aligned} & D(n) - D(n+1) \qquad (30) \\ =\ & (-G(\mathbf{f}(n+1))) - (-G(\mathbf{f}(n))) \\ \ge\ & \rho\|\mathbf{f}(n) - \mathbf{f}(n+1)\|^2 \end{aligned}$$

for some $\rho > 0$.

From (29) and (30),

$$D(n)^2 \le K_1^2 \|\mathbf{f}(n) - \mathbf{f}(n+1)\|^2 \le \frac{K_1^2}{\rho}(D(n) - D(n+1)),$$

or

$$D(n+1) \le D(n) - \frac{\rho}{K_1^2} D(n)^2.$$

As shown in [52],

$$\begin{aligned} \frac{1}{D(n+1)} &\ge \frac{1}{D(n)} \frac{1}{1 - \frac{\rho}{K_1^2} D(n)} \\ &= \frac{1}{D(n)} \sum_{i=0}^{\infty} \left(\frac{\rho}{K_1^2} D(n)\right)^i \\ &\ge \frac{1}{D(n)}\left(1 + \frac{\rho}{K_1^2} D(n)\right) = \frac{1}{D(n)} + \frac{\rho}{K_1^2}. \end{aligned}$$

Using induction, we have,

$$\frac{1}{D(n)} \ge \frac{1}{D(0)} + n\frac{\rho}{K_1^2},$$

or

$$D(n) \le \frac{1}{\frac{1}{D(0)} + n\frac{\rho}{K_1^2}} \le \frac{1}{n\frac{\rho}{K_1^2}}.$$

Thus,

$$G(\mathbf{f}(\infty)) - G(\mathbf{f}(n)) \le \frac{K_1^2}{\rho}\frac{1}{n}.$$

■

References

[1] K. Dell, "Welcome to wi-fi-ville," *Time*, Jan. 05, 2007.

[2] D. J. Baker, J. Wieselthier, and A. Ephremides, "A distributed algorithm for scheduling the activation of links in a self-organizing mobile radio network," in *IEEE ICC*, 1982, pp. 2F.6.1–2F.6.5.

[3] E. Arikan, "Some complexity results about packet radio networks," *IEEE Trans. on Info. Theory*, vol. 30, no. 4, pp. 681–685, Jul 1984.

[4] X. Lin and N. B. Shroff, "The impact of imperfect scheduling on crosslayer rate control in multihop wireless networks," in *Proc. Of INFOCOM*, 2005.

[5] A. Eryilmaz and R. Srikant, "Joint congestion control, routing, and MAC for stability and fairness in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1514 – 1524, August 2006.

[6] A. Kashyap, S. Sengupta, R. Bhatia, and M. Kodialam, “Two-phase routing, scheduling and power control for wireless mesh networks with variable traffic,” in *Sigmetrics07*, 2007.

[7] S. Sanghavi, L. Bui, and R. Srikant, “Distributed link scheduling with constant overhead,” in *SIGMETRICS*, 2007, pp. 313 – 324.

[8] S. Sarkar and L. Tassiulas, “End-to-end bandwidth guarantees through fair local spectrum share in wireless ad-hoc networks,” *IEEE Trans. on Automatic Control*, vol. 50, no. 9, pp. 1246–1259, Sep 2005.

[9] L. Bui, A. Eryilmaz, R. Srikant, and X. Wu, “Joint asynchronous congestion control and distributed scheduling,” in *Infocom*, 2006.

[10] B. Hajek and G. Sasaki, “Link scheduling in polynomial time,” *Information Theory, IEEE Transactions on*, vol. 34, no. 5, pp. 910–917, 1988.

[11] A. Brzenzinski, G. Zussman, and E. Modiano, “Enabling distributed throughput maximization in wireless mesh networks - a partitioning approach,” in *Mobicom*, 2006.

[12] P. Chaporkar, K. Kar, and S. Sarkar, “Throughput and fairness guarantees through maximal scheduling in wireless networks,” in *Proccedings of the Allerton Conference on Communication, Control, and Computing*, 2005.

[13] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, “Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks,” in *Infocom*, 2006.

[14] A. Behzad and I. Rubin, “On the performance of graph-based scheduling algorithms,” in *IEEE Globecom*, 2003, pp. 3432–3436.

[15] P. Wang and S. Bohacek, “Communication models for capacity optimization in mesh networks,” in *ACM PE-WASUN 2008*, Vancouver, Canada, 2008.

[16] ——, “On the practical complexity of solving the maximum weighted independent set problem for optimal scheduling in wireless networks,” University of Delaware, Tech. Rep., 2008.

[17] S. Bohacek, V. Sridhara, and J. Kim, “UDel Models,” available at: http://udelmodels.eecis.udel.edu/.

[18] F. Kelly, A. Maulloo, and D. Tan, “Rate control in communication networks: Shadow prices, proportional fairness and stability,” *Journal of the Operational Research Society*, vol. 49, pp. 237–252, Nov 1998.

[19] F. P. Kelly, “Charging and rate control for elastic traffic,” *European Transactions on Telecommunications*, vol. 8, pp. 33–37, January 1997.

[20] S. H. Low, “A duality model of TCP and queue management algorithms,” *IEEE/ACM Transactions on Networking*, vol. 11, pp. 525–536, August 2003.

[21] J. Mo and J. Walrand, “Fair end-to-end window-based congestion control,” *IEEE/ACM Transactions on Networking*, vol. 8, pp. 556–567, 2000.

[22] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu, “Impact of interference on multi-hop wireless network performance,” in *Proceedings of ACM MobiCom*, San Diego, CA, September 2003, pp. 66–80.

[23] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 2003.

[24] J. Nocedal and S. Wright, *Numerical Optimization*. Springer, 2000.

[25] T. ElBatt and A. Ephremides, “Joint scheduling and power control for wireless ad-hoc networks,” in *Proceedings of IEEE INFOCOM*, New York, NY, June 2002, pp. 976–985.

[26] M. Grotschel, L. Lovasz, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*. Berlin: Springer-Verlag, 1993.

[27] T. Matsui, “Approximation algorithms for maximum independent set problems and fractional coloring problems on unit disk graphs,” in *JCDCG*, 1998, pp. 194–200.

[28] G. Minty, “On maximal independent sets of vertices in claw-free graphs,” *J. Combinatorial Theory*, vol. B, no. 28, pp. 284–304, 1980.

[29] V. Alekseev, “A polynomial algorithm for finding the largest independent sets in fork-free graphs,” *Discrete Applied Mathematics*, vol. 135, pp. 3–16, 2004.

[30] G. H. Chen, M. T. Kuo, and J. P. Sheu, “An optimal time algorithm for finding a maximum weight independent set in a tree,” *BIT*, vol. 23, pp. 353–356, 1988.

[31] R. Karp and M. Sipser, “Maximum matchings in sparse random graphs,” in *FOCS*, 1981.

[32] G. Valiente, *A New Simple Algorithm for the Maximum-Weight Independent Set Problem on Circle Graphs*. Springer, 2003, vol. 2906, pp. 129–137.

[33] M. M. Halldórsson, *Approximation Algorithms for Combinatiorial Optimization*. Springer Berlin / Heidelberg, 2004, ch. Approximations of Independent Sets in Graphs, pp. 24–45.

[34] M. Fürer and S. P. Kasiviswanathan, “Algorithms for counting 2-SAT solutions and colorings with applications,” *Electronic Colloquium on Computational Complexity (ECCC)*, no. 033, 2005.

[35] F. V. Fomin, S. Gaspers, and S. Saurabh, “Branching and treewidth based exact algorithms,” in *ISAAC*, 2006, pp. 16–25.

[36] L. Babel, “A fast algorithm for the maximum weight clique problem,” *Computing*, vol. 52, pp. 31–38, 1994.

[37] E. Balas and J. Xue, “Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring,” *Algorithmica*, vol. 15, no. 5, pp. 397–412, 1996.

[38] ——, “Minimum weighted coloring of triangulated graphs, with application to maximum weight vertex packing and clique finding in arbitrary graphs,” *SIAM J. Comput.*, vol. 20, no. 2, pp. 209–221, 1991.

[39] J. S. Warren and I. V. Hicks, “Combinatorial branch-and-bound for the maximum weight independent set problem,” 2007.

[40] P. R. J. Östergård, “A fast algorithm for the maximum clique problem,” *Discrete Applied Mathematics*, vol. 120, no. 1-3, pp. 197 – 207, August 2002.

[41] H. N. Gabow, “Data structures for weighted matchings and nearest ancestors with linking,” in *ACM-SIAM Symposium on Discrete Algorithms*, 1990, pp. 434–443.

[42] Y.-S. Cheng, M. Neely, and K. M. Chugg, “Iterative message passing algorithm for bipartite maximum weighted matching,” in *2006 IEEE International Symposium on Information Theory*, 2006, pp. 1934–1938.

[43] R. Preis, “Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs,” in *Proceedings of The16th Annual Symposium on Theoretical Aspects of Computer Science (STACS 99)*, 1999.

[44] M. Hanckowiak, M. Karonski, and A. Panconesi, “A faster distributed algorithm for computing maximal matching deterministically,” in *Proceedings of PODC 99, the Eighteen Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, 1999, pp. 219–228.

[45] X. Lin and N. B. 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, April 2006.

[46] A. Czygrinow, M. Hanckowiak, and P. E. Szymanska, “Distributed algorithm for approximating the maximum matching,” *Discrete Applied Mathematics*, vol. 143, pp. 62–71, 2004.

[47] V. Sridhara and S. Bohacek, “Realistic propagation simulation of urban mesh networks,” *The International Journal of Computer and Telecommunications Networking Computer Networks and ISDN Systems (COMNET)*, 2007.

[48] T. Rappaport, *Wireless Communications - Principles and Practice*. Prentice Hall, 2002.

[49] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap*. Chapman and Hall/CRC, 1994.

[50] J. Zhu, X. Guo, L. Yang, and W. S. Conner, “Leveraging spatial reuse in 802.11 mesh networks with enhanced physical carrier sensing,” in *Proc. Of IEEE ICC*, 2004.

[51] K. C. Kapur, “On cutoff optimization methods in infinite-dimensional spaces and applications,” *JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS*, vol. 12, no. 1, pp. 16–31, 1973.

[52] D. M. Topkis, “A note on cutting-plane methods without nested constraint sets,” *Operations Research*, vol. 18, no. 6, pp. 1216–1220, 1970.