Approximate Aggregate Utility Maximization in Multi-Hop Wireless Networks using Distributed Greedy Scheduling

Albert Sunny, Joy Kuri*, Nachiket Sahasrabudhe

*Centre for Electronics Design and Technology, Indian Institute of Science, Bangalore-560012, India
Computational Research Laboratories, Pune-411030, India

Abstract
In this paper, we study the performance of greedy scheduling in multihop wireless networks, where the objective is aggregate utility maximization. Following standard approaches, we consider the dual of the original optimization problem. We note that the dual can be solved optimally, only with the knowledge of the maximal independent sets in the network. But computation of maximal independent sets is known to be NP-hard. Motivated by this, we propose a distributed greedy heuristic to address the problem of link scheduling. We evaluate the effect of the distributed greedy heuristic on aggregate utility maximization in detail, for the case of an arbitrary graph. We provide some insights into the factors affecting aggregate utility maximization in a network, by providing bounds on the maximum aggregate utility. We give simulation results for the approximate aggregate utility maximization achieved under distributed implementation of the greedy heuristic and find them close to the maximum aggregate utility obtained using optimal scheduling.

Key words: cross-layer optimization, distributed scheduling, \( \epsilon \)-subgradient, greedy scheduling

1. Introduction and Related Work

We consider a wireless multihop mesh network in which some nodes are sources of traffic. Each source node is assumed to have an infinite backlog of traffic that is to be sent to a destination node. Further, each source is equipped with a utility function that is concave increasing in the average data rate that it can push through the network. We are interested in obtaining a distributed cross-layer scheme for joint congestion control, routing and link scheduling, such that the aggregate utility is maximized.

Aggregate utility maximization seeks to strike a balance between high total throughput and fairness. At one extreme, a solution may maximize the sum of throughputs. However,
in such a solution, some sources may see very low throughputs even though the sum of throughputs is maximized. At the other extreme, a solution may be max-min fair, in which the smallest throughput is as large as possible, and, among such solutions, the second smallest is as large as possible, and so on. But, such a solution may lead to a low value of aggregate throughput. The aggregate utility maximization approach lies in between these extremes.

Research into scheduling, routing and congestion control is several decades old, but has seen a lot of activity following the seminal paper of Tassiulas and Ephremides. The literature can be classified into two broad groups. In the first group, traffic arrives into the network according to some specified random processes that cannot be controlled and the objective is to find the “capacity region” i.e., the largest set of arrival vectors for which a scheduling and routing policy can be found ensuring stable operation [1, 2, 3, 4, 5, 6].

In the second group, each source equipped with a utility function has an infinite backlog of data to send and the objective is to maximize aggregate utility. Following the important paper of Kelly, Maulloo and Tan [7], researchers have addressed the issue of obtaining distributed controls to achieve the objective. The basic idea is that the network will provide congestion signals to the sources (in the form of “prices”), and the sources will modify their data rates accordingly. The original problem is shown to decompose into several subproblems, viz., congestion control, routing and scheduling, and the objective is to find distributed solutions to each [8, 9, 10, 11, 12]. We consider a problem that belongs to the second group above.

This programme was later carried out for wireless networks, where the additional aspect of wireless link scheduling appeared [2, 13, 9, 14, 11, 15, 16, 17].

The main results of this paper can be summarized as follows:

- We develop a distributed greedy heuristic for the scheduling problem under the $K$-hop link interference model.
- We show that the distributed greedy heuristic leads to an $\epsilon$-subgradient which can be used to solve the problem of aggregate utility maximization.
- Further, we evaluate the effect of the sub-optimal greedy schedule by looking at the convergence properties of the $\epsilon$-subgradient method.
- We also provide some insights into the factors affecting aggregate utility maximization in a network, by providing bounds on the maximum aggregate utility.

2. System Model and Mathematical Formulation

2.1. Primal optimization problem

We assume the network to be a directed graph $G = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ represents the set of nodes in the network and $\mathcal{L}$ represents the set of wireless links in the network. As in [1], we assume that a bidirectional wireless link $(i, j) \in \mathcal{L}$ exists in the network, if nodes $i$ and $j$
are within transmission range of each other. We assume that each link $l \in \mathcal{L}$ has a capacity denoted by $C_l$. Let

$$C = \max_{l \in \mathcal{L}} C_l$$

i.e., $C$ represents the maximum link capacity in the network. Now, we can represent the capacity of a link $l \in \mathcal{L}$ as

$$C_l = \alpha_l \cdot C$$

where $0 < \alpha_l \leq 1$

The $\alpha_l$’s are dimensionless and we formulate the problem in terms of these dimensionless quantities. Let $\mathcal{F}$ denote the set of all end-to-end multihop flows present in the network. For each flow $f \in \mathcal{F}$, $s(f)$ and $d(f)$ represent the source and destination nodes of the flow $f$ respectively. We assume that the source nodes have an infinite backlog of data. Let $x_f$ denote the data rate associated with the flow $f$ and $y_{fl}$ denote the part of the flow $f$ that is carried by the link $l$. Let $y_f = (y_{fl})$, $l \in \mathcal{L}$ be a vector representing the part of flow $f$ carried by each link in the network.

In this paper, we model the inter-link interference using the $K$-hop link interference model. Here, we reproduce some definitions from [18] in order to define the interference model.

**Definition 1.** Let $d_S(x, y)$ denote the shortest distance (in terms of number of links) between nodes $x, y \in \mathcal{N}$. Define a function $d : (\mathcal{L}, \mathcal{L}) \to \mathbb{N}$ as follows: For links $l_u = (u_1, u_2), l_v = (v_1, v_2) \in \mathcal{L}$, let

$$d(l_u, l_v) = \min_{i,j \in \{1,2\}} d_S(u_i, v_j)$$

In the $K$-hop link interference model, we assume that any two links $l_1$ and $l_2$ for which $d(l_1, l_2) < K$, will interfere with each other and hence cannot be active simultaneously.

A maximal independent set of links ($\mathcal{I}$), is a set of links in which no two links of the set $\mathcal{I}$ interfere with each other under the given interference model and no other link can be added to the set $\mathcal{I}$ without violating the interference constraints. We represent a maximal independent set of links $\mathcal{I}$ by a column vector $\mathbf{r}_I$ of size $|\mathcal{L}|$. If a link $l \in \mathcal{I}$, then $\mathbf{r}_I(l)$ is $\alpha_l$; else it is 0. We represent the collection of all the maximal independent sets by the matrix $\mathbf{M}$, columns of which are the $\mathbf{r}_I$s. Let there be $J$ maximal independent sets present in the network.

We represent a schedule associated with them by a $J$ sized column vector $\mathbf{a}$, where the $i^{th}$ entry represents the fraction of time the independent set represented the $i^{th}$ column of the matrix $\mathbf{M}$ is active. We associate a strictly concave, twice differentiable, increasing utility function $U(x_f)$ with every end-to-end flow $f \in \mathcal{F}$. We also assume $U(0)$ to be less than or equal to zero. Such an assumption is natural, because it asserts that, if a source is not able to sustain a positive rate of data transfer, then the “utility” for that source is a finite non-positive value.

Let $\mathbf{A}$ be the $|\mathcal{N}| \times |\mathcal{L}|$ node-link incident matrix. Then, the formal representation of
the problem is as follows:

\[
\max_{x_f \geq 0, a \geq 0} \sum_{f \in F} U(x_f)
\]

Subject to: \(Ay_f = u_f, \forall f \in F\) \hspace{1cm} (1)

\(x_f \leq 1, \forall f \in F\) \hspace{1cm} (2)

\[\sum_{f \in F} y_f \leq Ma\] \hspace{1cm} (3)

\[\sum_{k=1}^{J} a_k = 1\] \hspace{1cm} (4)

where \(u_f\) represents a \(N\)-sized column vector such that \(u_f(s(f)) = x_f, u_f(d(f)) = -x_f\) and all other entries are zero. Here, the first constraint ensures flow conservation at every node in the network. The second constraint ensures that the data rates of the flows are less than or equal to the maximum link capacity in the network. The next constraint, represented by Equation (3) ensures that the aggregate flow on each link is less than or equal to the effective capacity of the link. The fourth constraint ensures that there are no idle slots in the schedule.

2.2. Dual problem

The primal optimization problem is a convex optimization problem with affine constraints. By applying Slater’s condition [19], it can be shown that this problem has no duality gap. To obtain a solution in a distributed manner, we consider the dual problem as in [7, 8]. The capacity constraints given by Equation (3) are relaxed to obtain the Lagrange variables. These Lagrange variables behave as link prices and we represent them by a vector \(p\). Now, the dual problem associated with the primal problem can be stated as follows:

\[
\min_{p \geq 0} D(p)
\]

where

\[
D(p) = \max_{x_f \geq 0, a \geq 0} \left( \sum_{f \in F} (U(x_f) - p^T (y_f - Ma)) \right)
\]

Subject to: \(Ay_f = u_f, \forall f \in F\)

\(x_f \leq 1, \forall f \in F\)

\[\sum_{k=1}^{J} a_k = 1\]
We note that given a price vector $\mathbf{p}$, the RHS of Equation (5) can be written as a sum of two functions of price vector $\mathbf{p}$ i.e.,

$$ D(\mathbf{p}) = D_1(\mathbf{p}) + D_2(\mathbf{p}) $$

Here, $D_1(\mathbf{p})$ corresponds to the congestion control and routing problem, whereas $D_2(\mathbf{p})$ represents the link scheduling problem.

3. Congestion Control and Routing Subproblem

$$ D_1(\mathbf{p}) = \max_{x_f \geq 0} \sum_{f \in \mathcal{F}} \left( U(x_f) - \sum_{i=1}^{\mathcal{L}} p_i y_{fi} \right) $$

Subject to:

$$ A y_f = u_f, \forall f \in \mathcal{F} $$
$$ x_f \leq 1, \forall f \in \mathcal{F} $$

For a given vector of link prices $\mathbf{p}$, each source solves the problem of how much traffic to send into the network so as to maximize its net utility. Since the maximum value that $x_f$ can attain is 1, we can obtain the optimal value of $x_f$ as

$$ x_f = \min\{U^{-1}(p(f)), 1\} \quad (6) $$

where $p(f)$ is cost of the least-priced path between $s(f)$ and $d(f)$ for a given $\mathbf{p}$. Since all the link capacities are normalized with respect to $C$, the $x_f$ obtained from Equation (6) represents the data rate normalized with respect to $C$. Hence, the actual rate at which traffic is to be injected into the network is given by $x_f \cdot C$. The least-priced path for a flow $f$ can be found by using a modified Distributed Bellman-Ford algorithm, which uses the link price as the metric in place of hop count. If there are multiple least-priced paths, then the traffic can be split among them in any arbitrary manner.

4. Scheduling Subproblem

$$ D_2(\mathbf{p}) = \max_{a \geq 0} \mathbf{p}^t \mathbf{c} $$

Subject to: $c = \sum_f r_f a_f$ and $\sum_f a_f = 1$

It can be shown that for a given vector of link prices $\mathbf{p}$, the solution to this problem is to schedule an independent set of “maximum aggregate capacity-weighted price” [19] i.e., a
maximal independent set of links $\mathcal{I}_{\text{opt}}$ is the optimal solution if

$$\mathcal{I}_{\text{opt}} = \arg \max_{\mathcal{I}} \left( \sum_{l \in \mathcal{I}} \alpha_l p_l \right)$$

Thus, an optimal scheduling vector $a_{\text{opt}}(p)$ has all its entries zero, except the one corresponding to an independent set with maximum aggregate “capacity-weighted price”. Computing the optimal schedule requires knowledge of all the maximal independent sets in the network and is known to be NP-hard [18]. Hence, it is difficult to realize the same in a distributed manner for an adhoc network. We employ a distributed greedy heuristic to obtain a distributed solution for the links scheduling problem. We denote the schedule obtained using the distributed greedy heuristic by $a_{\text{dgrd}}(p)$.

4.1. Distributed Greedy Heuristic

In this section, we consider a centralized greedy heuristic and examine the possibility of implementing the same in a distributed manner. Here, we present the Greedy Heuristic.

**Algorithm 1 Centralized Greedy Heuristic**

1: Set $W := \emptyset$ and $i := 1$.
2: Arrange the links of $L$ in descending order of capacity-weighed price (i.e., $\alpha_l p_l$), starting with $l_1, l_2, \ldots$
3: If $W \cup l_i$ is a valid $K$-matching, then, $W := W \cup l_i$, $i = i + 1$.
4: Repeat Step 3 for all links in $L$.

Here, a set of edges $W$ is a valid $K$-matching if $\forall l_1, l_2 \in W$ with $l_1 \neq l_2$, we have $d(l_1, l_2) \geq K$.

In [20], the authors have shown that the Greedy Maximal Schedule (GMS) achieves the full capacity region in tree networks under the $K$-hop interference model and that the worst-case efficiency ratio of GMS in geometric unit-disc graphs is between $\frac{1}{6}$ and $\frac{1}{3}$.

Next, we present a distributed version of the greedy heuristic. The algorithm for the distributed greedy heuristic is described below and is implemented at every node $n \in \mathcal{N}$.

**Algorithm 1 Distributed Greedy Heuristic**

The algorithm is described in terms of the messages exchanged between nodes in each slot of the $m^{th}$ ROUND and the decisions that are made at the slot boundaries after the completion of the given slot. Together, the SEND LINK PRICES slot, the SEND MARKED LINKS slot and the SEND STATUS slot constitute a ROUND as shown in Figure 1.

![Figure 1: Slot division of the distributed greedy algorithm.](image-url)
Each link $l$ can have four possible states namely OPEN ($O$), CHECK ($CH$), MARKED ($M$) and CLOSED ($CL$). A link $l \in \mathcal{L}$ is called an “attached link” of node $n \in \mathcal{N}$, if node $n$ is an end point of the link $l$ and the link $l$ is directed outwards from node $n$. Initially, all links are set to OPEN, the algorithm status set is to DO NOT TERMINATE and $m$ is set as 1.

Algorithm 2 Pseudo-code for Distributed Greedy Heuristic

\textbf{In slot $S^m_L$}

1: Find the highest capacity-weighed priced (i.e., $\alpha_l p_l$) link among all the OPEN attached links and disseminate this information over the ($K+1$)-hop neighbourhood.

\textbf{At time $T^m_L$}

1: \textbf{if} at least one attached link is OPEN \textbf{then}
2: \hspace{1em} sort the OPEN attached links in descending order of their respective capacity-weighed price. Let $l'_{\text{max}}$ be the maximum capacity-weighed priced link among the attached OPEN links.
3: \hspace{1em} \textbf{if} no capacity-weighed prices are received from the (K+1)-hop neighbours \textbf{then}
4: \hspace{2em} link $l'_{\text{max}}$ is MARKED and all other OPEN attached links are CLOSED and go to 17.
5: \hspace{1em} \textbf{else}
6: \hspace{2em} sort the received OPEN links in descending order of their respective capacity-weighed price. Let $l_{\text{max}}$ be the maximum capacity-weighed priced link among the received OPEN links.
7: \hspace{2em} \textbf{end if}
8: \hspace{2em} \textbf{if} ($p_{l'_{\text{max}}} > p_{l_{\text{max}}}$) \textbf{then}
9: \hspace{3em} link $l'_{\text{max}}$ is MARKED and all other OPEN attached links are CLOSED.
10: \hspace{2em} \textbf{else}
11: \hspace{3em} for all OPEN attached link $l$, do
12: \hspace{4em} \textbf{if} ($d(l, l_{\text{max}}) < K$) \textbf{then}
13: \hspace{5em} link $l$ is set to CHECK.
14: \hspace{4em} \textbf{end if}
15: \hspace{3em} \textbf{end for}
16: \hspace{2em} \textbf{end if}
17: \hspace{1em} \textbf{end if}

The links that are MARKED are the maximum capacity-weighed priced links in their corresponding ($K+1$) hop neighbourhoods. Also, the links that are moved to the CLOSED state are certain to have a MARKED link within $K$ hop link distance. If a link moves to MARKED or CLOSED state in ROUND $m$, it will continue to remain in that state until the algorithm terminates (i.e., these states are absorbing states). These links will not participate in price dissemination in the subsequent ROUNDs. Links that are in the MARKED state will be scheduled, upon termination of the algorithm. The reason for introducing a CHECK
state is to help resolve ambiguities that arise due to decisions based on information from local neighbourhood.

Algorithm 2: Pseudo-code for Distributed Greedy Heuristic

In slot $S^m_M$

1: if any of the attached links is MARKED then
2: disseminate this information to the $(K + 1)$-hop neighbourhood.
3: end if

At time $T^m_M$

1: for each attached link $l$ in state CHECK, do
2: if $(d(l, \text{received MARKED link}) < K)$ for at least one received MARKED link then
3: link $l$ is CLOSED.
4: else
5: link $l$ remains in CHECK state.
6: end if
7: end for
8: OPEN the highest capacity-weighed priced attached CHECK link.
9: Algorithm status is set to TERMINATE at nodes which have no OPEN or CHECK links.

During the price dissemination slot, links will move into CHECK state if they see a higher-capacity-weighed priced interfering link, but are unable to decide if such a link will get MARKED. In the SEND MARKED LINKS slot, CHECK links get to know if there is indeed a higher capacity-weighed priced MARKED link interfering with it. If so, the link in CHECK state is CLOSED.

Algorithm 2: Pseudo-code for Distributed Greedy Heuristic

In slot $S^m_T$

1: if at least one attached link is OPEN or in CHECK then
2: send a DO NOT TERMINATE message to all nodes in the $(K+1)$-hop neighbourhood.
3: else if a DO NOT TERMINATE message is received then
4: send a DO NOT TERMINATE message to all nodes in the $(K + 1)$ hop neighbourhood.
5: end if

At time $T^m_T$

1: if no DO NOT TERMINATE message is received then
2: the algorithm has terminated, schedule all MARKED links.
3: else
4: go to the $(m + 1)^{th}$ ROUND.
5: end if
The local termination condition is that no attached link is in OPEN or CHECK state. In the above slot, this information is conveyed to all the other nodes in the network in a distributed manner. This makes sure that the algorithm terminates in a synchronous fashion at each node.

Next, we illustrate the working of our distributed greedy heuristic under a 2-hop link interference model with a couple of examples. First, let us consider a linear network with 7 nodes as shown in Figure 2. For data transfer, only links (1,2), (2,3), (3,4), (4,5), (5,6) and (6,7) are considered; but control traffic can flow in the opposite direction too. In Table 1, we show the link states against the time when different decisions are made.

| T  | 1.0 | 0.9 | 0.8 | 0.7 | 0.6 | 0.5 |
|----|-----|-----|-----|-----|-----|-----|
| (1,2) | O   | O   | O   | O   | O   | O   |
| (2,3) | M   | CH  | CH  | CH  | CH  | CH  |
| (3,4) | M   | CL  | CL  | O   | O   | O   |
| (4,5) | M   | CL  | CL  | M   | CH  | CH  |
| (5,6) | M   | CL  | CL  | M   | CL  | CL  |

Figure 2: An example to illustrate the distributed greedy scheduling algorithm.

Table 1: Table showing the states of the links in Figure 2 against the various slots of the distributed greedy scheduling algorithm.

In the first ROUND, only link (1,2) is MARKED. All other links see a higher capacity-weighted priced interfering link and thus move into the CHECK state. Then, link (1,2) announces it is MARKED. Upon reception of this information, links (2,3) and (3,4) are CLOSED, since they interfere with link (1,2). But all other links are moved to OPEN, since they do not find any interfering MARKED link. This process repeats until the network has no OPEN or CHECK links. The set of links that are scheduled after the algorithm terminates, are shown in bold, in Figure 3.

| 1  | 2  | 3  | 4  | 5  | 6  | 7  |
|----|----|----|----|----|----|----|
| 1.0 | 0.9 | 0.8 | 0.7 | 0.6 | 0.5 | 0.5 |

Figure 3: Figure showing scheduled links (in bold) using distributed greedy scheduling on the example in Figure 2.

Next, we consider a 7 node linear network shown in Figure 4. As in the previous example, we show the link states against the time when different decisions are made in Table 2.

In this example, the highest capacity-weighted priced link is located in the middle of the network. As a result, more links are moved into the CLOSED state after the first ROUND.
The only remaining OPEN link, i.e., link (1,2), is MARKED in the subsequent ROUND. The set of links that are scheduled after the algorithm terminates, are shown in bold, in Figure 5.

One can compute the centralized greedy schedule for the above examples easily and verify that the distributed greedy schedule matches it.

### 4.2. Performance of Greedy Scheduling

In this section, we show analytically that the distributed greedy heuristic schedules the same set of links as the centralized greedy heuristic. These results are from our previous work [21], but are included here for completeness.

**Lemma 1.** The algorithm terminates in finite time.

**Proof.** In Appendix A.

Let us assume that no two links have equal capacity-weighted prices i.e., $\forall i, j \in \mathcal{L}, p_i \alpha_i \neq p_j \alpha_j$; or in other words, we have a unique way of breaking tie among equal capacity-weighted priced links.

**Lemma 2.** For every link $i$ CLOSED before ROUND $m$, there must be at least one link MARKED before ROUND $m$, that interferes with it.

**Proof.** In Appendix A.
The additional CHECK state helps to resolve ambiguities that arise in deciding the highest capacity-weighted priced link based on information from a local neighbourhood. If a link \( i \) sees an interfering higher capacity-weighted priced link in its neighbourhood, it will move into CHECK state. Such a link, will subsequently move to the CLOSED state, if it receives an announcement to CLOSE from an interfering higher capacity-weighted priced MARKED link. Now, we would like to formally show that the introduction of CHECK state does not alter the performance of the algorithm.

**Lemma 3.** At the beginning of a ROUND, consider the globally highest capacity-weighted priced link among links that are neither CLOSED nor MARKED; such a link will not be in CHECK state. Or, it can be restated as; at the end of a round, the highest capacity-weighted priced CHECK link is OPENed.

**Proof.** In Appendix A.

**Theorem 1.** The distributed greedy algorithm and the centralized greedy algorithm schedule the same links.

**Proof.** In Appendix A.

Next, we quantify the deviation of our distributed greedy schedule from the optimal one.

**Definition 2.** [18] The \( K \)-hop interference set of a link \( l \in \mathcal{L} \), denoted by \( \mathcal{I}_K(l, \mathcal{L}) \), is the set of links \( m \in \mathcal{L} \) such that \( d(l, m) < K \).

We call a subset \( \mathcal{D} \) of \( \mathcal{I}_K(l, \mathcal{L}) \) “\( K \)-maximal”, if no link \( m \in \mathcal{I}_K(l, \mathcal{L}) \setminus \mathcal{D} \) can be added to \( \mathcal{D} \), without violating the interference constraints. In other words, for all \( m \in \mathcal{I}_K(l, \mathcal{L}) \) and all \( n \in \mathcal{D} \), \( d(m, n) < K \).

**Definition 3.** [18] The \( K \)-hop interference degree of a link \( l \in \mathcal{L} \), denoted by \( d_K(l) \), is defined as

\[
d_K(l) = \max_{S \subseteq \mathcal{I}_K(l, \mathcal{L}); \mathcal{D} \text{is } K\text{-maximal}} |\mathcal{D}|
\]

**Definition 4.** [18] The \( K \)-hop interference degree of a graph \( G(\mathcal{N}, \mathcal{L}) \), denoted by \( d_K(G) \), is defined as

\[
d_K(G) = \max_{l \in \mathcal{L}} d_K(l)
\]

For a given price vector \( \mathbf{p} \), let \( S_{\text{opt}}(\mathbf{p}) \), \( S_{\text{grd}}(\mathbf{p}) \) and \( S_{\text{dgrd}}(\mathbf{p}) \) represent the aggregate capacity-weighted prices of the independent sets corresponding to the optimal schedule, greedy schedule and distributed greedy schedule, respectively. From Theorem 1 we know that \( S_{\text{dgrd}}(\mathbf{p}) = S_{\text{grd}}(\mathbf{p}) \). Substituting this in a theorem from [18], we have

**Theorem 2.**

\[
\frac{S_{\text{opt}}(\mathbf{p})}{S_{\text{dgrd}}(\mathbf{p})} \leq d_K(G)
\]

Further, there exists a graph \( G \) for which the above ratio is exactly \( d_K(G) \).

**Proof.** We refer to Theorem 5 in [18].
5. $\epsilon$-Subgradient Method

In order to solve the dual problem, we have to minimize the dual objective function $D(p)$. The natural strategy would be to consider the negative gradient and take a small step in that direction. However, we notice that $D(p)$ is not differentiable in $p$ everywhere. This raises the possibility of using subgradients.

**Lemma 4.** $D(p)$ is a convex function of $p$.

**Proof.** In Appendix B.

**Definition 5.** [22] Given a convex function $D(p) : \mathcal{R}^n \rightarrow \mathcal{R}$, we say that a vector $h(p) \in \mathcal{R}^n$ is an $\epsilon$-subgradient of $D(p)$ at point $p \in \mathcal{R}^n$, if

$$D(p) \geq D(p) - \epsilon + (p - p)^T h(p), \quad \forall p \in \mathcal{R}^n.$$

**Proposition 1.** For the given price vector $p$, the price updation rule leads to an $\epsilon(p)$-subgradient $h(p)$ associated with price vector $p$, given by

$$h(p) = c_{\text{dgrd}}(p) - y(p) \quad \text{and} \quad \epsilon(p) = p^T (c_{\text{opt}}(p) - c_{\text{dgrd}}(p))$$

where the $l^{th}$ entry of $c_{\text{dgrd}}(p)$ is $\alpha_l$ if link $l$ is present in the distributed greedy schedule $a_{\text{dgrd}}(p)$ associated with the price vector $p$, else it is zero; the $l^{th}$ entry of $c_{\text{opt}}(p)$ is $\alpha_l$ if link $l$ is present in the optimal schedule $a_{\text{opt}}(p)$ associated with the price vector $p$, else it is zero; further, the $l^{th}$ entry of $y(p)$ represents the aggregate flow on link $l \in \mathcal{L}$ when the price vector is $p$.

**Proof.** In Appendix B.

The $\epsilon$-subgradient method can be used to generate a sequence of dual feasible points according to the iteration [22]

$$p[j + 1] = (p[j] - \delta \cdot h(p[j]))^+$$

Here, $(x)^+ = \max(x, 0)$ and $\delta$ is the constant step size associated with the subgradient algorithm. Hence, we update the price of link $l$ going from the $j^{th}$ to the $(j + 1)^{th}$ iteration using the following equation:

$$p_l[j + 1] = (p_l[j] + \delta (y_l[j] - c_l[j]))^+ \quad (7)$$

where $y_l[j]$ represents the aggregate flow on link $l$ in the $j^{th}$ iteration obtained from the optimal solution of the routing and congestion control problem. If link $l$ is part of the distributed greedy schedule for the given price vector $p[j]$ in the $j^{th}$ iteration, then $c_l[j]$ takes the value $\alpha_l$, else it is zero.

Next, we provide a theorem that helps us establish a bound on the deviation of the approximately maximized aggregate utility from the maximum aggregate utility achievable. Since it has been shown in [10], that the subgradient method with constant step size results in oscillations in “steady-state” (limit cycles), we only look at the convergence of the dual in the cesaro sense and not in the strict sense.
Theorem 3. If \( \lim_{j \to \infty} \epsilon(p[j]) \leq \epsilon \) and \( \| h(p[j]) \|_2 \leq H < \infty \) \( \forall j \), then

\[
D(p^*) \leq \lim_{j \to \infty} \frac{1}{j} \sum_{\tau=0}^{j-1} D(p[\tau]) \leq D(p^*) + \frac{\delta H^2}{2} + \epsilon
\]

where \( p^* \) is an optimal solution to the dual problem \( D(p) \).

Proof. In Appendix B.

Theorem 3 tells us that the average of the dual function obtained using the distributed greedy algorithm lies in a band above the optimal dual value.

6. Conclusion and Future Work

The scheduling problem is known to be a bottleneck in the cross-layer optimization approach. We have relax the optimality requirement and proposed a distributed greedy heuristic that schedules an independent set of links. Further, we have also quantified the effect of the sub-optimal distributed greedy heuristic by established. Even though we bound the performance of the distributed greedy scheduling in Theorem 3, it seems that one can obtain a tighter bounds atleast for specific type of networks. We would also like to address this problem in our future work.

References

[1] L. Tassiulas and A. Ephremides, “Stability properties of constrained queuing systems and scheduling policies for maximum throughput in multihop radio networks,” in IEEE Transactions on Automatic Control, vol.37, no.12, pp. 1936-1948, August 1992.

[2] A. Stolyar, “Maximizing queuing network utility subject to stability: greedy primal-dual algorithm,” in Queueing Systems vol.50, no.4, pp.401-457, 2005.

[3] R. Buche and H.J. Kushner, “Control of mobile communication systems with time-varying channels,” in IEEE Transactions on Automatic Control, vol.49, no.11, pp. 1954 - 1962, November 2004.

[4] S. Shakkottai and A. Stolyar, “Scheduling of a shared time varying channel: the exponential rule stability,” in INFORMS Applied Probability Conference, New York, July 2001.

[5] ——, “Scheduling for multiple flows sharing a time varying channel: The exponential rule,” in Transactions of AMS, 2001.

[6] A. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, P. Whiting, and K. Vijaykumar, “Providing quality of service over a shared wireless link,” in IEEE Communications Magazine vol.39, pp.150-153, February 2001.

[7] F.P. Kelly, A. Maullo, and D. Tan, “Rate control in communication networks: shadow prices, proportional fairness and stability,” in J. Oper. Res. Soc., vol. 49, 1998, pp. 237–252.

[8] L. Chen, S. Low, and J. Doyle, “Joint congestion control and media access control design for ad hoc wireless networks,” in Proceedings of IEEE Infocom, March 2005.

[9] 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 Proceedings of IEEE Infocom, 2006.

[10] F.L. Presti, “Joint congestion control, routing and media access control optimization via dual decomposition for ad hoc wireless networks,” in Proceedings of the 8-th International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), Montreal, Canada, October 2005.
Let $L^m_O$ be the set of OPEN links before ROUND $m$. Let $L^m_C$ be the set of CLOSED links before ROUND $m$. Let $L^m_H$ be the set of CHECK links before ROUND $m$. Let $L^m_M$ be the set of MARKED links before ROUND $m$.

**Proof of Lemma 7**

**Proof.** If the set $L^m_O$ is null, the algorithm terminates. Hence, we note that there is at least one OPEN link at the beginning of each round. Let

$$l^m = \arg \max_{l \in L^m_O} p_l \alpha_l$$

be the global maximum-capacity-weighted priced link before ROUND $m$. Since the capacity-weighted price of this link among all the OPEN attached links is the highest, it is also the local maximum among the OPEN attached links received from the ($K+1$)-hop neighborhood. Thus link $l^m$ gets MARKED. Since a MARKED link will always remain in the same state,

$$l \in L^m_M, \text{ then } l \in L^k_M, \forall k \geq m + 1 \implies L^m_M \subseteq L^{m+1}_M$$
Now let us consider link \( l^m \),
\[
l^m \in \mathcal{L}_O^m \implies l^m \notin \mathcal{L}_M^m
\]
But from the previous argument, link \( l^m \) gets \textit{MARKED} in \textit{ROUND} \( m + 1 \). Thus
\[
l^m \in \mathcal{L}_M^{m+1} \text{ and } l^m \notin \mathcal{L}_M^m \implies \mathcal{L}_M^m \subset \mathcal{L}_M^{m+1}
\] (8)
Since a link \textit{CLOSED} in \textit{ROUND} \( m \), will remain \textit{CLOSED} for the subsequent \textit{ROUNDS}, we have
\[
\text{if } l \in \mathcal{L}_C^m, \text{ then } l \in \mathcal{L}_C^k, \forall k \geq m + 1 \implies \mathcal{L}_C^m \subseteq \mathcal{L}_C^{m+1}
\] (9)
Now from (8) and (9) we have
\[
\mathcal{L}_C^m \cup \mathcal{L}_M^m \subset \mathcal{L}_C^{m+1} \cup \mathcal{L}_M^{m+1}
\]
At all times, a link \( l \) can be in one of the four states, i.e.,
\[
\forall m, \mathcal{L}_C^m \cup \mathcal{L}_M^m \cup \mathcal{L}_O^m \cup \mathcal{L}_H^m = \mathcal{L}
\]
From the above two argument,
\[
\mathcal{L}_O^{m+1} \cup \mathcal{L}_H^{m+1} \subset \mathcal{L}_O^m \cup \mathcal{L}_H^m
\]
Since the number of links in set \( \mathcal{L} \) is finite, there exists a \( t < \infty \), such that
\[
\mathcal{L}_O^t \cup \mathcal{L}_H^t = \{\phi\}
\]
Thus the algorithm terminates in finite number of \textit{ROUNDS} and thus in finite time. \( \square \)

\textbf{Proof of Lemma 2}

\textbf{Proof.} Assume that there is no such \( j \in \mathcal{L}_M^m \) for some \( i \in \mathcal{L}_C^m \). Then the link \( i \) would not have received any \textit{MARKED} link that interferes with it, in slot \( S_{m-1}^m \) (Algorithm 2, At time \( T_{m}^m \), lines 5, 6). Then, this would imply that either link \( i \) would be \textit{OPENed} or would be in \textit{CHECK}. i.e.,
\[
i \in \mathcal{L}_H^m \cup \mathcal{L}_O^m \implies i \notin \mathcal{L}_C^m
\]
But this is a contradiction. Thus there exists a link \( j \in \mathcal{L}_M^m \), such that \( d(i, j) < K \). \( \square \)

\textbf{Proof of Lemma 3}

\textbf{Proof.} Let us assume that there is no \( j \in \mathcal{L}_O^m \) for some \( i \in \mathcal{L}_H^m \), such that \( \alpha_{i,p_i} < \alpha_{j,p_j} \). This would imply that \( \alpha_{i,p_i} > \alpha_{j,p_j}, \forall j \in \mathcal{L}_O^m \). Since link \( i \in \mathcal{L}_H^m \), we can say that a link \( k \), such that \( \alpha_{k,p_k} > \alpha_{i,p_i} \) and \( d(k, i) = 0 \) was \textit{OPENed} at time \( T_{m-1}^m \) (Algorithm 2, At time \( T_{m-1}^m \), line 8). Since link \( k \) was \textit{OPENed} at time \( T_{m-1}^m \), \( k \in \mathcal{L}_O^m \). Let
\[
\alpha_{i,p_i} > \alpha_{j,p_j}, \forall j \in \mathcal{L}_O^m
\]
But we have $\alpha_k p_k > \alpha_i p_i$. Thus

$$\alpha_k p_k > \alpha_j p_j, \forall j \in L_O^m \implies k \notin L_O^m$$

But this is a contradiction to the statement that $k \in L_O^m$. Thus $\forall i \in L_H^m, \alpha_i p_i < \alpha_j p_j$, for some $j \in L_O^m$. Let

$$\alpha_{p_m} = \max_{l \in L \setminus (L_H^m \cup L_C^m)} \alpha_{p_l}$$

$$i = \arg \max_{l \in L_H^m} \alpha_{p_l}$$

It is evident that $i \in L_H^m$. Thus from the previous claim, there exists a $j \in L_O^m$ such that

$$\alpha_j p_j > \alpha_i p_i$$

i.e.,

$$\alpha_j p_j > \max_{l \in L_H^m} \alpha_{p_l}$$

Also

$$\max_{k \in L_O^m} \alpha_k p_k \geq \alpha_j p_j > \max_{l \in L_H^m} \alpha_{p_l}$$

Hence

$$\alpha_{p_m} = \max_{k \in L_O^m} \alpha_k p_k$$

Thus $\arg \alpha_{p_m} \in L_O^m$. \hfill \Box

**Proof of Theorem 1**

**Proof.** Let $L_C$ be the set of links CHOSEN by the centralized greedy algorithm. Let the set $L_C$ be ordered and indexed in the decreasing order of link capacity-weighted price as \{l_1, l_2, ..., l_v\}. Let the distributed greedy algorithm terminate after $t$ ROUNDs. Let $L_{M}^{t+1}$ be the set of MARKED links after the termination of the algorithm.

We need to prove that every link CHOSEN by the centralized greedy algorithm is MARKED by the distributed greedy algorithm, by the time it terminates i.e., $L_C \subseteq L_{M}^{t+1}$. We will prove the above claim via induction.

**Induction statement:** If links $l_1, l_2, ..., l_k \in L_C$ then $l_1, l_2, ..., l_k \in L_{M}^{t+1}$.

**Basis:** To show the statement holds for the globally maximum capacity-weighted priced link.

Let link $l_1 \in L_C$ be the globally maximum capacity-weighted priced link. Thus, this link will also be a local maximum among interfering links in a $(K + 1)$-hop neighbourhood. Hence, this link will be MARKED after the $1^{st}$ ROUND i.e., $l_1 \in L_{M}^{1}$. Since MARKED is
an absorbing state, \( l_1 \in \mathcal{L}_{M,1}^{t+1} \). Now, let us define
\[
\mathcal{I}(y) = \{ l \in \mathcal{L} : d(l, y) < K \}
\]
as the set of links that interfere with link \( y \). Let
\[
\mathcal{L}_{k+1} = \mathcal{L} - \bigcup_{i=1}^{k} (l_i \cup \mathcal{I}(l_i))
\]
be the set of links left after links \( \{l_1, l_2, ..., l_k\} \) are CHOSEN. Let
\[
\forall l \in \mathcal{L}_{k+1}, \mathcal{P}(l) = \{ l' \in \mathcal{L}_{k+1} : d(l, l') < K, \alpha_{l'} p_{l'} > \alpha_l p_l \} \tag{10}
\]
It is obvious that for link \( L_{k+1} \) to be CHOSEN, \( \mathcal{P}(l_{k+1}) = \{ \phi \} \).

Inductive step: If \( l_1, l_2, ..., l_k \in \mathcal{L}_{M,1}^{t+1} \) given that \( l_1, l_2, ..., l_k \in \mathcal{L}_C \), then if \( l_{k+1} \in \mathcal{L}_C \) then \( l_{k+1} \in \mathcal{L}_M^{t+1} \). Since \( L_{1}, L_{2}, ..., L_{k} \in \mathcal{L}_M^{t+1} \), for each \( i \in \{1, 2, ..., k\} \)
\[
\exists m_i \leq t + 1 : l_i \in \mathcal{L}_M^{m_i}, l_i \notin \mathcal{L}_M \text{ for } s < m_i
\]
Since MARKED is an absorbing state,
\[
l_i \in \mathcal{L}_M^{r}, \ \forall m_i \leq r \leq t + 1
\]
Let
\[
\forall l \in \mathcal{L}_O^{m}, \mathcal{P}^m(l) = \{ l' \in \mathcal{L}_O^{m} : d(l, l') < K, \alpha_{l'} p_{l'} > \alpha_l p_l \}
\]
We note that link \( l \) is MARKED in ROUND \( m \), if \( \mathcal{P}^m(l) = \{ \phi \} \). Let
\[
m' = \max_{i=1}^{k} m_i
\]
It is easy to see that before the end of ROUND \( m' \), links \( \{l_1, l_2, ..., l_k\} \) are MARKED and the links that interfere with these links are CLOSED. Thus
\[
\mathcal{L}_O^{m'} \subseteq \mathcal{L} - \bigcup_{i=1}^{k} (l_i \cup \mathcal{I}(l_i)) \subseteq \mathcal{L}_{k+1}^{t+1} \tag{11}
\]
Now,
\[
\mathcal{P}^{m'}(l_{k+1}) = \{ l' \in \mathcal{L}_O^{m'} : d(l_{k+1}, l') < K, \alpha_{l'} p_{l'} > \alpha_{l_{k+1}} p_{l_{k+1}} \} \ \forall l_{k+1} \in \mathcal{L}_O^{m'}
\]
From (10) and (11), we can say that
\[
\mathcal{P}^{m'}(l_{k+1}) \subseteq \mathcal{P}(l_{k+1})
\]
Since \( l_{k+1} \in \mathcal{L}_C \), \( \mathcal{P}(l_{k+1}) = \{ \phi \} \). Thus
\[
\mathcal{P}^{m'}(l_{k+1}) \subseteq \{ \phi \} \implies \mathcal{P}^{m'}(l_{k+1}) = \{ \phi \}
\]
Now, since the algorithm terminates after \( t \) \text{ROUND}s,
\[
\mathcal{L}_O^{t+1} \cup \mathcal{L}_H^{t+1} = \{ \phi \}
\]
\[
\therefore l_{k+1} \notin \mathcal{L}_O^{t+1} \implies m' \neq t + 1
\]
Thus link \( l_{k+1} \) gets \text{MARKED} in \text{ROUND} \( m' \leq t \). Thus link \( l_{k+1} \) gets \text{MARKED} before the algorithm terminates. Therefore,
\[
\forall l \in \mathcal{L}_C, \; l \in \mathcal{L}_M^{t+1} \implies \mathcal{L}_C \subseteq \mathcal{L}_M^{t+1}
\]
Now, let us assume that LHS is a strict subset of RHS, i.e.,
\[
\mathcal{L}_C \subset \mathcal{L}_M^{t+1}
\]
Then there exists a link \( l_i \) such that \( l_i \notin \mathcal{L}_C \) but \( l_i \in \mathcal{L}_M^{t+1} \). Since \( l_i \in \mathcal{L}_M^{t+1} \), we can say that
\[
d(l, l_i) \geq K, \; \forall l \in \mathcal{L}_M^{t+1}
\]
Since \( \mathcal{L}_C \subset \mathcal{L}_M^{t+1} \),
\[
d(l, l_i) \geq K, \; \forall l \in \mathcal{L}_C
\]
If the above was true, then \( l_i \in \mathcal{L}_C \). But this contradicts our assumption that \( \mathcal{L}_C \subset \mathcal{L}_M^{t+1} \).
Thus LHS can not be a strict subset of RHS.
\[
\implies \mathcal{L}_C = \mathcal{L}_M^{t+1}
\]
\[\square\]

Appendix B

Proof of Proposition 7

PROOF. Let \( p_1, p_2 \in \mathcal{R}_+^L \). Let \( x_f(p) \) and \( y_f(p) \) represent the optimal flow rate and optimal routing vector of a flow \( f \in \mathcal{F} \) respectively. \( y(p) \) represents a \( L \) sized column vector, \( l^{th} \) entry of which indicates the aggregate traffic of all the flows carried by the link \( l \) for the
price vector \( p \), i.e., \[ \sum_{f \in F} y_f(p) \]. Then at \( p_2 \) we have the following:

\[
D(p_2) = \sum_{f \in F} U(x_f(p_2)) - p_2^T(y(p_2) - c_{opt}(p_2)) \\
\geq \sum_{f \in F} U(x_f(p_1)) - p_2^T(y(p_1) - c_{dgrd}(p_1)) \\
- (p_2 - p_1)^T(y(p_1) - c_{dgrd}(p_1)) \\
- p_1^T(c_{opt}(p_1) - c_{dgrd}(p_1)) \\
\geq D(p_1) + (p_2 - p_1)^T(c_{dgrd}(p_1) - y(p_1)) \\
- p_1^T(c_{opt}(p_1) - c_{dgrd}(p_1)) \tag{12}
\]

From the definition of the \( \epsilon \)-subgradient, we have

\[
D(p_2) \geq D(p_1) - \epsilon + (p_2 - p_1)^T h(p_1) \tag{13}
\]

By comparing Equation 12 and Equation 13 we can write

\[ h(p) = c_{dgrd}(p) - y(p) \]

Also note that \( p_1^T(c_{dgrd}(p_1) - c_{opt}(p_1)) \) is always a positive quantity, thus we can write

\[ \epsilon(p) = p^T(c_{opt}(p) - c_{dgrd}(p)) \]

\[ \square \]

**Proof of Theorem 3**

**PROOF.** Let \( p^* \) be the optimal price vector of the dual problem. From the price update equation, we have

\[ p[j + 1] = (p[j] + \delta(y[j] - c[j]))^+ \]

Equivalently,

\[
||p[j + 1] - p^*||_2^2 \leq ||p[j] + \delta(y[j] - c[j]) - p^*||_2^2 \\
= ||p[j] - p^*||_2^2 + \delta^2||y[j] - c[j]||_2^2 + 2\delta(p[j] - p^*)^T(y[j] - c[j]) \tag{14}
\]

From the definition of an \( \epsilon \)-subgradient, we have

\[ D(\bar{p}) \geq D(p) - \epsilon + (\bar{p} - p)^T h(p), \quad \forall \bar{p} \in \mathbb{R}^n \]

Hence, we have

\[
(p[j] - p^*)^T(y[j] - c[j]) \leq -(D(p[j]) - D(p^*) + \epsilon(p[j])) \tag{15}
\]
Henceforth, we represent $\epsilon(p[j])$ as $\epsilon_j$ for convenience. Substituting Equation (15) in Equation (14), we get

$$||p[j + 1] - p^*||_2^2 \leq ||p[j] - p^*||_2^2 + \delta^2 ||y[j] - c[j]||_2^2 - 2\delta(D(p[j]) - D(p^*) + \epsilon_j)$$

Applying the inequalities recursively, we obtain

$$||p[j + 1] - p^*||_2^2 \leq ||p[1] - p^*||_2^2 + \delta^2 \sum_{\tau=1}^{j} ||y[\tau] - c[\tau]||_2^2 - 2\delta(\sum_{\tau=1}^{j}(D(p[\tau]) - D(p^*))) + \sum_{\tau=1}^{j} \epsilon_{\tau})$$

Since $||p[j + 1] - p^*||_2^2 \geq 0$, $\forall j$, we get

$$2\delta \sum_{\tau=1}^{j}(D(p[\tau]) - D(p^*)) \leq ||p[1] - p^*||_2^2 + \delta^2 \sum_{\tau=1}^{j} ||y[\tau] - c[\tau]||_2^2 + \sum_{\tau=1}^{j} \epsilon_{\tau}$$

Since we assume that $||y[\tau] - c[\tau]||_2 = \|h(p[\tau])\| \leq H$, we have

$$\limsup_{j \to \infty} \frac{1}{j} \sum_{\tau=0}^{j-1}(D(p[\tau]) - D(p^*)) \leq \frac{\delta H^2}{2} + \limsup_{j \to \infty} \frac{1}{j} \sum_{\tau=0}^{j-1} \epsilon_{\tau}$$

Since we assume that $\lim_{j \to \infty} \epsilon_j \leq \epsilon$, there exists $j_0$ such that $\epsilon_j \leq \epsilon$, $\forall j \geq j_0$. Therefore, for large enough $j$, we have

$$\limsup_{j \to \infty} \frac{1}{j} \sum_{\tau=1}^{j} \epsilon_{\tau} \leq \limsup_{j \to \infty} \frac{1}{j} \sum_{\tau=1}^{j_0} \epsilon_{\tau} + \limsup_{j \to \infty} \frac{(j - j_0)}{j} \epsilon = \epsilon$$

□