Throughput-Delay Trade-off for Aggregated Convergecast in Large-Scale Wireless Sensor Networks

Amitabha Ghosh

Autonomous Networks Research Group
Ming Hsieh Department of Electrical Engineering
University of Southern California, Los Angeles
amitabhg@usc.edu
http://anrg.usc.edu/~amitabhg

August 12, 2009
Outline

1. Maximizing Convergecast Throughput
   - Motivation and Preliminaries
   - Scheduling with Unlimited Frequencies
   - Scheduling with Limited Number of Frequencies

2. Improving Throughput-Delay Trade-off
   - Motivation
   - Bi-criteria Formulation
   - Optimal Spanning Tree
1 Maximizing Convergecast Throughput
   - Preliminaries
   - Scheduling with Unlimited Frequencies
   - Scheduling with Limited Number of Frequencies

2 Improving Throughput-Delay Trade-off
   - Motivation
   - Bicriteria Formulation
   - Optimal Spanning Tree
Model and Assumptions
Model and Assumptions

- Graph-based network & interference model

E.g., Monitoring applications with fast & efficient delivery requirements.
Model and Assumptions

- Graph-based network & interference model
- Half-duplex, single transceiver

E.g., Monitoring applications with fast & efficient delivery requirements.
Model and Assumptions

- Graph-based network & interference model
- Half-duplex, single transceiver
- Receiver-based Frequency Assignment

E.g., Monitoring applications with fast & efficient delivery requirements.
Model and Assumptions

- Graph-based network & interference model
- Half-duplex, single transceiver
- Receiver-based Frequency Assignment
- Non-interfering orthogonal channels

E.g., Monitoring applications with fast & efficient delivery requirements.
Model and Assumptions

- Graph-based network & interference model
- Half-duplex, single transceiver
- Receiver-based Frequency Assignment
- Non-interfering orthogonal channels
- TDMA, periodic scheduling
  (- every node generates a single packet at the beginning of each frame
  - every node transmits during only a single slot)

E.g., Monitoring applications with fast & efficient delivery requirements.
Model and Assumptions

- Graph-based network & interference model
- Half-duplex, single transceiver
- Receiver-based Frequency Assignment
- Non-interfering orthogonal channels
- TDMA, periodic scheduling
  (- every node generates a single packet at the beginning of each frame
  - every node transmits during only a single slot)
- In-network aggregation
  (distributive, algebraic
  - aggregation happens at each hop)

E.g., Monitoring applications with fast & efficient delivery requirements.
**Convergecast:**

A many-to-one communication pattern (i.e., flow of data): from all the nodes to a sink (opposite of broadcast).

---

*Figure: Single frequency.*
**Convergecast:**

A many-to-one communication pattern (i.e., flow of data): from all the nodes to a sink (opposite of broadcast).

![Diagram of a network with nodes a, b, c, d, e, f, g and the sink s, illustrating the flow of data with frequencies f1.](image)

**Figure:** Schedule length is 5.
**Convergecast:**

A many-to-one communication pattern (i.e., flow of data): from all the nodes to a sink (opposite of broadcast).

**Figure:** Schedule length is 5.

**Figure:** Two frequencies.
Convergecast:

A many-to-one communication pattern (i.e., flow of data): from all the nodes to a sink (opposite of broadcast).

Figure: Schedule length is 5.

Figure: Schedule length is 3.
**Convergecast:**

A many-to-one communication pattern (i.e., flow of data): from all the nodes to a sink (opposite of broadcast).

<table>
<thead>
<tr>
<th>Receiver</th>
<th>Frame 1</th>
<th>Frame 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Slot 1</td>
<td>Slot 2</td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure: Schedule length is 3.
**Convergecast:**

A many-to-one communication pattern (i.e., flow of data): from all the nodes to a sink (opposite of broadcast).

<table>
<thead>
<tr>
<th>Receiver</th>
<th>Frame 1</th>
<th>Frame 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Slot 1</td>
<td>Slot 2</td>
</tr>
<tr>
<td>s</td>
<td>c</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>e</td>
<td>f</td>
</tr>
<tr>
<td>c</td>
<td>g</td>
<td></td>
</tr>
</tbody>
</table>

**Figure:** Schedule length is 3.
Problem Formulation

**Given:**

A set of nodes, arbitrarily deployed (possibly even worst case), a set of frequencies, and a routing (spanning) tree rooted at the sink.

**Define:**

Aggregated Throughput = 1/Schedule Length

**Joint Frequency and Time Slot Assignment Problem:**

Assign a frequency to each receiver and a time slot to each edge in the routing tree, such that the aggregated throughput at the sink is maximized (equivalent to minimizing the schedule length).
Our Approach

Two Subproblems:
Split the *Joint Frequency and Time Slot Assignment* problem into two subproblems:

- Frequency Assignment subproblem
- Time Slot Assignment subproblem

Two Scenarios:
- Scheduling with *unlimited* frequencies
- Scheduling with a *limited* number of frequencies
Scenario One: Scheduling with Unlimited Frequencies
**Question:**

Given a spanning tree $T$ on an *arbitrary graph* $G = (V, E)$, can we find the *minimum number of frequencies* that can be assigned to the receivers of $T$ such that *all* the secondary interfering links are removed? *(Minimum Frequency Assignment problem)*
Question:
Given a spanning tree $T$ on an arbitrary graph $G = (V, E)$, can we find the minimum number of frequencies that can be assigned to the receivers of $T$ such that all the secondary interfering links are removed? (Minimum Frequency Assignment problem)

Theorem
Minimum Frequency Assignment is NP-hard on general graphs.
Proof: Reduction from Vertex Color.
Construction of Constraint Graph:

- For each receiver in $G$ create a node in $G_C$.
- Create an edge between two nodes in $G_C$ if the corresponding receivers in $G$ form secondary interfering links.
Construction of Constraint Graph:

- For each receiver in $G$ create a node in $G_C$.
- Create an edge between two nodes in $G_C$ if the corresponding receivers in $G$ form secondary interfering links.

Lemma:

$$K_{max} \leq \Delta(G_C) + 1,$$
where $\Delta(G_C)$: max degree in $G_C$. (Vertex Color)
Construction of Constraint Graph:

- For each receiver in $G$ create a node in $G_C$.
- Create an edge between two nodes in $G_C$ if the corresponding receivers in $G$ form secondary interfering links.

Lemma:

$$K_{\text{max}} \leq \Delta(G_C) + 1,$$

where $\Delta(G_C)$: max degree in $G_C$. (Vertex Color)
**Parameters:**

- Number of nodes: $n : 200$, square region of area $A : 200 \times 200$
- Shortest Path Tree, Unit Disk Graph
- Density: $n/A$

**Figure:** Comparison of the upper bound with the Largest Degree First scheme.
Maximizing Convergecast Throughput
Improving Throughput-Delay Trade-off

Preliminaries
Scheduling with Unlimited Frequencies
Scheduling with Limited Number of Frequencies

Time Slot Assignment Subproblem
Algorithm: **BFS-TimeslotAssignment**

1. **while** $E_T \neq \emptyset$ **do**
2. \[ e \leftarrow \text{next edge from } E_T \text{ in BFS order} \]
3. Assign minimum time slot to \( e \) respecting adjacency constraint
4. \( E_T \leftarrow E_T \setminus \{e\} \)
5. **end while**

---

Diagram:

**G**

- $s$ to $f_3$ with length 2
- $f_3$ to $f_1$ and $f_1$ to $b$ with length 3

Schedule length: 3 (minimum)
Time Slot Assignment Subproblem

Algorithm: **BFS-TimeSlotAssignment**

1. while $E_T \neq \emptyset$ do
2. \hspace{1em} $e \leftarrow$ next edge from $E_T$ in BFS order
3. \hspace{1em} Assign minimum time slot to $e$ respecting adjacency constraint
4. \hspace{1em} $E_T \leftarrow E_T \setminus \{e\}$
5. end while

**Theorem**

*BFS-TimeSlotAssignment* gives a schedule of minimum length $\Delta(T)$, where $\Delta(T)$: maximum node degree in the routing tree $T$.

*Proof:* By induction.
Scenario Two: Scheduling with a Limited Number of Frequencies
Scheduling Complexity
Question:
Given a spanning tree $T$ on an arbitrary graph $G$, is there an assignment of time slots using at most $q$ frequencies such that the schedule length is at most $p$?
(Multi-Frequency Scheduling Problem)
**Question:**

Given a spanning tree $T$ on an arbitrary graph $G$, is there an assignment of time slots using *at most* $q$ *frequencies* such that the *schedule length is at most* $p$?

(*Multi-Frequency Scheduling Problem*)

**Theorem**

*Muti-Frequency Scheduling* is NP-complete on general graphs.

*Proof:* Reduction from *Vertex Color.*
Conjecture:
The Joint Frequency and Time Slot Scheduling Problem is NP-hard even under the Disk Graph models, e.g., Unit Disk Graphs. (Future Work)

Question:
Can we design approximation algorithms (i.e., frequency and time slot assignment schemes) that have provably good worst case bounds on unit disk graphs?
Approximation Results on Unit Disk Graph
Approximation Results on Unit Disk Graph

**Algorithm $G$: Two Phases**

1. Phase 1: Assign a frequency to each receiver in each cell.
2. Phase 2: Assign a time slot greedily to each edge. E.g., schedule a *maximal number of edges* in each time slot.
Phase 1: Frequency Assignment

**Goal:**
Assign the frequencies to the receivers in such a way that the maximum number of nodes (load) transmitting on the same frequency is minimized. (Load-Balanced Frequency Assignment)
Phase 1: Frequency Assignment

Goal:
Assign the frequencies to the receivers in such a way that the maximum number of nodes (load) transmitting on the same frequency is minimized. (Load-Balanced Frequency Assignment)

Lemma:
Load-Balanced Frequency Assignment is NP-hard.
Phase 1: Frequency Assignment

**Algorithm:** \texttt{FrequencyGreedy}

1. Sort the receivers in non-increasing order of in-degrees:
   \[ \text{deg}^{in}(v_1) \geq \text{deg}^{in}(v_2) \geq \ldots \geq \text{deg}^{in}(v_n) \]

2. Starting from \( v_1 \), assign each successive receiver a frequency that has the \textit{least load}, breaking ties arbitrarily.
phase 1: frequency assignment

**Lemma:**

**FrequencyGreedy** gives a \((\frac{4}{3} - \frac{1}{3K})\) approximation on the min-max load, where \(K\) is the number of frequencies.

**Proof:** Follows from Graham’s list scheduling according to Longest Processing Time first [Graham ’69].
Phase 2: Time Slot Assignment

\[ |\gamma_c| \leq 2( T_{v_2} + 2 ) \]
Lemma:

Suppose $L_{c}^{FG}$ denote the load on the maximally loaded frequency in cell $c$ achieved by \textsc{FrequencyGreedy}. Then, any greedy time slot assignment can schedule all the edges in $c$ within number of time slots bounded by: $|\gamma_c| \leq 2 \cdot L_{c}^{FG}$, where $\gamma_c$ denotes the set of time slots to schedule all the edges in $c$. 

|$\gamma_c| \leq 2 (T_{v_2} + 2)$
Phase 2: Time Slot Assignment
Phase 2: Time Slot Assignment
Lemma:

The \textit{minimum schedule length} for the whole network is upper bounded by:

\[ 4 \cdot \max_c \left\{ |\gamma_c| \right\}, \forall \alpha \geq 2 \]

Proof: The colors represent sets of disjoint time slots. Follows from \textit{Vertex Color} on rectangular grid.
Approximation Bound on Unit Disk Graph

Algorithm $G$

1. Run $\text{FrequencyGreedy}$ in each cell.
2. Run any greedy time slot assignment scheme.
   e.g., schedule a maximal number of edges in each slot.
Maximizing Convergecast Throughput
Improving Throughput-Delay Trade-off

Preliminaries
Scheduling with Unlimited Frequencies
Scheduling with Limited Number of Frequencies

Approximation Bound on Unit Disk Graph

Algorithm $G$

1. Run \textsc{FrequencyGreedy} in each cell.
2. Run any greedy time slot assignment scheme.
   e.g., schedule a \textit{maximal number of edges} in each slot.

Theorem

Algorithm $G$ achieves a constant factor $8\mu_\alpha \cdot \left(\frac{4}{3} - \frac{1}{3K}\right)$ approximation on the optimal schedule length, where $\mu_\alpha > 0$ is a constant for a given cell size $\alpha \geq 2$ and for a given deployment of nodes.

($\mu_\alpha$: maximum number of edges on the same frequency in any cell that can be scheduled on the same time slot)
Approximation Bound on Unit Disk Graph

**Proof Sketch:**

Lower bound on $OPT$:

$$OPT \geq \frac{1}{\mu \alpha} \max_c \left\{ L_c^{LB} \right\}$$

Schedule length of $G$:

$$\Gamma_G \leq 4 \cdot \max_c \left\{ |\gamma_c| \right\}$$

$$\leq 4 \cdot \max_c \left\{ 2 \cdot L_c^{FG} \right\}$$

$$\leq 8 \cdot \max_c \left\{ \left( \frac{4}{3} - \frac{1}{3K} \right) \cdot L_c^{LB} \right\}$$

$$\leq 8\mu \alpha \left( \frac{4}{3} - \frac{1}{3K} \right) \cdot OPT$$
Performance Evaluation

Parameters:

- Square region $A : 200 \times 200$, Shortest Path Tree, Unit Disk Graph

Figure: Schedule length of Algorithm $G$ with network size for different number of frequencies $K$. Schedule length bottlenecked by the in-degree of the parents beyond a certain number of frequencies.
Assignment on General Disk Graphs
Assignment on General Disk Graphs

Notations:

- \( r(u) \): Transmission range of node \( u \)
- \( \ell(e) \): Length of edge \( e \)
- \( I(e) \): Set of edges that are either adjacent to \( e \) or form a secondary interfering link with \( e \)
- \( I_\geq(e) \): Subset of \( I(e) \) whose end points have larger disks than \( e \)

\[
I_\geq(e) = \{ e' = (u', v') : e' \in I(e), \ell(e') \geq \ell(e) \}
\]
ILP Formulation of Frequency Assignment

0-1 ILP:

- **Goal**: Minimize the maximum number of edges that interfere with any given edge.

- Define indicator variables $X_{vk}$ for edge $e = (u, v)$ as:

  $X_{vk} = \begin{cases} 
  1, & \text{if } v \text{ is assigned frequency } f_k \\
  0, & \text{otherwise}
  \end{cases}$

A frequency assignment is therefore a 0-1 assignment to $X_{vk}, \forall e, \forall f_k$. 
**ILP Formulation of Frequency Assignment**

**0-1 ILP:**

- **Goal:** Minimize the maximum number of edges that interfere with any given edge.
- Define indicator variables $X_{vk}$ for edge $e = (u, v)$ as:

$$X_{vk} = \begin{cases} 
1, & \text{if } v \text{ is assigned frequency } f_k \\
0, & \text{otherwise} 
\end{cases}$$

A frequency assignment is therefore a 0-1 assignment to $X_{vk}, \forall e, \forall f_k$.

- For every $e = (u, v)$ on $f_k$, define $Z_{ek}$ as the total number of edges in $I_{\geq}(e)$ that are also on $f_k$

$$Z_{ek} = \sum_{e' = (u', v') \in I_{\geq}(e)} X_{v'k} = \sum_{v'} n(e, v')X_{v'k},$$

where $n(e, v') = |\{e' \in I_{\geq}(e) : \ell(e') \geq \ell(e)\}|$
0-1 ILP:

\[
\begin{align*}
\text{Minimize} & \quad \lambda \\
\text{subject to} : & \\
\forall e = (u, v), \forall f_k : & \quad \sum_{v'} n(e, v') X_{vk} \leq \lambda \quad (1) \\
\forall e = (u, v) : & \quad \sum_{f_k} X_{vk} = 1, \quad (2) \\
\forall e = (u, v), \forall f_k : & \quad X_{vk} \in \{0, 1\} \quad (3)
\end{align*}
\]

- Solve the Linear Relaxation (LP) by modifying constraint (3)
- **Randomized Rounding**: Assign \( Y_{vk} = 1 \) with probability \( X_{vk}^* \), where \( X_{vk}^* \) are the optimal fractional LP solutions and \( Y_{vk} \) are the new integral random variables.
Lemma:

Let $Y_{vk}$ be the rounded solution, as described above. Then,

$$\max_{e,f_k} \left\{ \sum_{e'=(u',v') \in I \geq (e)} Y_{v'k} \right\} = O\left( \Delta(T) \log n \cdot \lambda^* \right),$$

with probability at least $(1 - 1/n)$.

Proof: Apply Chernoff bound.
Lemma:

Let $X_{vk}$ and $Y_{vk}$ be defined as above. Then, $\Omega(\max_{e,f_i}\{Z_{ek}\})$ is a lower bound on the length of any schedule for the edges. Also, it is possible to schedule all the edges using $\max_{e,f_i}\{Z_{ek}\}$ time slots.
Lemma:
Let $X_{vk}$ and $Y_{vk}$ be defined as above. Then, $\Omega(\max_{e,f_i}\{Z_{ek}\})$ is a lower bound on the length of any schedule for the edges. Also, it is possible to schedule all the edges using $\max_{e,f_i}\{Z_{ek}\}$ time slots.

Time Slot Assignment:

- Sort the edges in non-increasing order of lengths: $\ell(e_1) \geq \ell(e_2) \ldots$
- Assign the smallest available slot $t(e_i)$ to edge $e_i$ such that:
  - $t(e_i) \neq t(e_j)$, $\forall e_j$ with $j < i$ having the same receiver as $e_i$
  - $t(e_i) \neq t(e_j)$, $\forall e_j$ that are assigned the same frequency as $e_i$
**Lemma:**

Let $X_{vk}$ and $Y_{vk}$ be defined as above. Then, $\Omega\left(\max_{e,f_i}\{Z_{ek}\}\right)$ is a lower bound on the length of any schedule for the edges. Also, it is possible to schedule all the edges using $\max_{e,f_i}\{Z_{ek}\}$ time slots.

**Time Slot Assignment:**

- Sort the edges in non-increasing order of lengths: $\ell(e_1) \geq \ell(e_2) \ldots$
- Assign the smallest available slot $t(e_i)$ to edge $e_i$ such that:
  - $t(e_i) \neq t(e_j)$, $\forall e_j$ with $j < i$ having the same receiver as $e_i$
  - $t(e_i) \neq t(e_j)$, $\forall e_j$ that are assigned the same frequency as $e_i$

**Theorem**

The schedule constructed by time slot assignment strategy along with the frequency assignment using the above randomized rounding procedure results in a schedule of length $O(\Delta(T) \log n)$ times the optimum.
Outline

1. Maximizing Convergecast Throughput
   - Preliminaries
   - Scheduling with Unlimited Frequencies
   - Scheduling with Limited Number of Frequencies

2. Improving Throughput-Delay Trade-off
   - Motivation
   - Bicriteria Formulation
   - Optimal Spanning Tree
Properties of Different Spanning Trees

**Shortest Path Tree:**
*(shallow and fat)*

- High node degree $\Rightarrow$ low throughput
- Fewer hops $\Rightarrow$ low delay
Properties of Different Spanning Trees

Shortest Path Tree: (shallow and fat)
- High node degree ⇒ low throughput
- Fewer hops ⇒ low delay

Minimum Interference Tree (weighted MST) ([Burkhart, Mobihoc ’04]): (deep and skinny)
\[ w(u, v) : \# \text{ of nodes covered by the union of} \]
\[ \text{the two disks centered at } u, v, \text{ each of radius } |uv| \]
- Low node degree ⇒ high throughput
- More hops ⇒ high delay
Scheduling Performance on Spanning Trees

Figure: Comparison of schedule length with network size for SPT and MIT.
Scheduling Performance on Spanning Trees

Figure: Comparison of schedule length with network size for SPT and MIT.

Questions:
- What are the desired properties of a spanning tree?
- Is constant factor approximation on schedule length and delay still achievable?
Theorem

If the maximum node degree of the routing tree is bounded by a constant $\Delta_C > 0$, then there exists an algorithm $H$ that achieves a constant factor $8\mu_\alpha \cdot \Delta_C$ approximation on the optimal schedule length, where $\mu_\alpha > 0$ is a constant for a given cell size $\alpha \geq 2$ and for a given deployment.
Tree Property for Bounded Throughput

Theorem

If the maximum node degree of the routing tree is bounded by a constant $\Delta_C > 0$, then there exists an algorithm $\mathcal{H}$ that achieves a constant factor $8\mu_\alpha \cdot \Delta_C$ approximation on the optimal schedule length, where $\mu_\alpha > 0$ is a constant for a given cell size $\alpha \geq 2$ and for a given deployment.

Properties:

- Degree-bounded spanning trees?
- Constructing a Minimum Degree Spanning Tree is NP-hard on general graphs [Furer, Raghavachari, SODA’92]. What about on UDG?
- What about bounding the delay?
Bicriteria Optimization Problem
Bicriteria Optimization Problem

**Definition:** [Cheriyan, Ravi, ’98]

Is a problem \( \mathcal{P} = (\mathcal{X}, \mathcal{Y}, \mathcal{H}) \), where \( \mathcal{X} \) and \( \mathcal{Y} \) are two objectives, and \( \mathcal{H} \) represents a subgraph.

- \( c_\mathcal{X} \): Cost function w.r.t. \( \mathcal{X} \)
- \( c_\mathcal{Y} \): Cost function w.r.t. \( \mathcal{Y} \)
- \( B_{\mathcal{X}} \): Budget on \( \mathcal{X} \) under \( c_\mathcal{X} \)

**Goal:** Find an \( \mathcal{H} \) that minimizes \( \mathcal{Y} \) under \( c_\mathcal{Y} \) subject to \( B_{\mathcal{X}} \)

Hard problems, especially when the objectives are *hostile* w.r.t. each other.
Bicriteria Optimization Problem

**Definition:** [Cheriyan, Ravi, '98]

Is a problem \( P = (\mathcal{X}, \mathcal{Y}, \mathcal{H}) \), where \( \mathcal{X} \) and \( \mathcal{Y} \) are two objectives, and \( \mathcal{H} \) represents a subgraph.

- \( c_{\mathcal{X}} \): Cost function w.r.t. \( \mathcal{X} \)
- \( c_{\mathcal{Y}} \): Cost function w.r.t. \( \mathcal{Y} \)
- \( B_{\mathcal{X}} \): Budget on \( \mathcal{X} \) under \( c_{\mathcal{X}} \)

**Goal:** Find an \( H \) that minimizes \( \mathcal{Y} \) under \( c_{\mathcal{Y}} \) subject to \( B_{\mathcal{X}} \)

Hard problems, especially when the objectives are *hostile* w.r.t. each other.

**Formulation:** Bounded-Degree Minimum-Radius Spanning Tree:

\( P = (\text{Degree, Radius, Spanning Tree}) \)

Radius: \( R(T) = \text{Maximum hop distance from any node to the sink in } T \)

\( \Delta^* = \text{Budget on max degree} \)

**Goal:** Find a spanning tree of minimum radius subject to max degree \( \Delta^* \)
Bicriteria Approximation

**Definition:**
An algorithm $A$ is an $(\alpha, \beta)$-approximation to the bicriteria optimization problem $\mathcal{P} = (\text{Degree, Radius, Spanning Tree})$ if:

- $\Delta(T) \leq \alpha + \Delta^*$
- $R(T) \leq \beta \cdot R^*(T)$

where $R^*(T)$ is the minimum radius of any spanning tree $T$ whose maximum node degree is $\Delta^*$. 
Definition:

An algorithm $A$ is an $(\alpha, \beta)$-approximation to the bicriteria optimization problem $\mathcal{P} = (\text{Degree, Radius, SpanningTree})$ if:

- $\Delta(T) \leq \alpha + \Delta^*$
- $R(T) \leq \beta \cdot R^*(T)$

where $R^*(T)$ is the minimum radius of any spanning tree $T$ whose maximum node degree is $\Delta^*$.

Characteristics of Bicriteria Formulation: ([Marathe, et al. JoA '08])

- **Robust**: Any $(\alpha, \beta)$-approximation on $\mathcal{P} = (\text{Degree, Radius, SpanningTree})$ can be transformed in polynomial time into a $(\beta, \alpha)$-approximation on $\mathcal{P}' = (\text{Radius, Degree, SpanningTree})$.

- **General**: Subsumes the cases of functional combinations of the two objectives.
Routing Tree Construction

**Algorithm:** $BA$

$BA$ produces a **Bounded-Degree Minimum-Radius Spanning Tree**

Three phases:

1. Construct a **Global Backbone Tree** of low radius.
2. Construct degree-bounded **Local Spanning Trees**.
3. **Merge** the local spanning trees with the global backbone tree.
Backbone Tree Construction

Complete graph

1/2
Backbone Tree Construction
Backbone Tree Construction

- **Sink**
  - (Initialize the backbone tree $T_B$ with the sink)
Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
  (Mark the cell containing the sink active, all other cells inactive)
Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
  (Choose any node, a local root, from a neighboring inactive cell)
Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
  (Connect the local root to $T_B$)
Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
  (Mark the cell containing the local root active)
Backbone Tree Construction

- Sink
- Active cell
- Inactive cell
- Local root

(Choose another local root from a neighboring inactive cell)
Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**

(Choose an intermediary node that could connect the local root to $T_B$)
Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**

(Connect the intermediary node and the local root to $T_B$)
Maximizing Convergecast Throughput

Improving Throughput-Delay Trade-off

Motivation

Bicriteria Formulation

Optimal Spanning Tree

Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
  (Mark the cell containing the local root active)
Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
Maximizing Convergecast Throughput
Improving Throughput-Delay Trade-off

Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
Backbone Tree Construction

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
Backbone Tree Construction

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
Backbone Tree Construction

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
Maximizing Convergecast Throughput
Improving Throughput-Delay Trade-off

Motivation
Bicriteria Formulation
Optimal Spanning Tree

Backbone Tree Construction

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
Backbone Tree Construction

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
Backbone Tree Construction

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
Backbone Tree Construction

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
Backbone Tree Construction

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
Maximizing Convergecast Throughput
Improving Throughput-Delay Trade-off

Motivation
Bicriteria Formulation
Optimal Spanning Tree

Backbone Tree Construction

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
Backbone Tree Construction

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
Backbone Tree Construction

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
Backbone Tree Construction

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
Maximizing Convergecast Throughput
Improving Throughput-Delay Trade-off

Motivation
Bicriteria Formulation
Optimal Spanning Tree

Backbone Tree Construction

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
Backbone Tree Construction

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
Local Spanning Tree

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
Local Spanning Tree

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
- Nodes in local spanning tree $T_c$
  (Nodes within each cell form a complete graph)
Local Spanning Tree

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
- **Nodes in local spanning tree**
  (Connect the nodes to form $T_c$ with max degree $\Delta^*$)
Local Spanning Tree

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
- Nodes in local spanning tree (Connect the nodes to form $T_c$ with max degree $\Delta^*$)
Local Spanning Tree

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
- Nodes in local spanning tree
  (Connect the nodes to form $T_c$ with max degree $\Delta^*$)
Local Spanning Tree

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
- **Nodes in local spanning tree**
Local Spanning Tree

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
- Nodes in local spanning tree
Local Spanning Tree

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
- Nodes in local spanning tree
Local Spanning Tree

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
- **Nodes in local spanning tree**
Local Spanning Tree

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
- Nodes in local spanning tree
Local Spanning Tree

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
- **Nodes in local spanning tree**
Local Spanning Tree

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
- Nodes in local spanning tree
Local Spanning Tree

- **Sink**
- **Active cell**
- **Inactive cell**
- **Local root**
- **Intermediary node**
- **Nodes in local spanning tree**
Local Spanning Tree

- Sink
- Active cell
- Inactive cell
- Local root
- Intermediary node
- Nodes in local spanning tree
Algorithm \( BA \) achieves a \((\alpha, \beta)\) approximation for the Bounded-Degree Minimum-Radius Spanning Tree problem, where \( \alpha > 0 \) and \( \beta > 0 \) are constants.
Performance Evaluation: Schedule Length and Delay

*Comparison of schedule length and delay with different network sizes for three types of trees.*
Conclusions and Future Work

- Studied the throughput-delay trade-off for aggregated convergecast
- Proved NP-completeness of the joint frequency and time slot assignment problem using multiple frequencies.
- Designed constant factor approximation for minimizing the schedule length for unit disk graphs.
- Designed $\Delta(T) \log n$ approximation for minimizing the schedule length general disk graphs.
- Bi-criteria formulation of the throughput-delay trade-off problem.
- Designed constant factor approximation for degree-bounded minimum-radius spanning trees.
- Extend the above work for SINR based models.