Weekly Report (2010-02-21)

Hongxing Li

In past two weeks, I were mainly working on three tasks.

1) Marking the assignment 1 of CSIS0234B.
2) Trying to solve our research problem with primal-dual decomposition.
3) Designing a distributed data aggregation algorithm integrated with successive interference cancellation.

I. Primal-Dual Decomposition

The main obstacle in the way to a distributed solution is the requirement of global information for SINR constraint. So we have to relax the SINR constraint by bounding the maximum possible interference from current transmitters with different receivers, which are separated by a distance of r_c as defined in previous report between each other. However, the solution for this relaxed problem is not the optimal solution for the original optimization problem.

In order to guarantee that each current receiver will be separated by a distance of r_c, we need to bring in some additional constraints on each receiver j.

$$s_{jit} + \sum_{u} \sum_{v \in V_i^{(I)}} s_{uvt} \leq 1, \ \forall i, j, u \in [1, n]$$

$$\sum_{j \notin V_i^{(T)}} s_{ijt} = 0, \ \forall i \in [1, n-1]$$

here, $V_i^{(I)}$ and $V_i^{(T)}$ are the sets of nodes that are in the range of r_c and in the range of d_M, which is the maximum transmission range defined in previous report, from node i respectively.

Suppose the bounded interference is I. It can be calculated in the same way in previous report. Then we can

convert the optimization problem into

$$
\begin{aligned}
minimize \quad & a\sum_{t} y_t + b\sum_{i,j,t} P_{ijt} \\
s.t. \quad & \sum_{j,t} s_{ijt} = 1,\ \forall i \in [1, n-1] \\
& \sum_{j,t} s_{njt} = 0 \\
& \sum_{j} s_{ijt} \leq 1 - \frac{1}{n^2}\sum_{l \geq t,j} s_{jil},\ \forall i,t \in [1,n] \\
& s_{jit} + \sum_{u}\sum_{v \in V_i^{(I)}} s_{uvt} \leq 1,\ \forall i,j,u \in [1,n] \\
& \sum_{j \notin V_i^{(T)}} s_{ijt} = 0,\ \forall i \in [1, n-1] \\
& \sum_{i,j} s_{ijt} \leq n^2 y_t,\ \forall t \in [1,n] \\
& 0 \leq P_{ijt} \leq P_{max}, \quad \forall t,i,j \in [1,n] \\
& G_{ij}P_{ijt} - \beta\sum_{u>i} G_{uj}P_{ujt} - \beta I - \beta N_0 \geq \Phi(s_{ijt} - 1), \quad \forall t,i,j \in [1,n] \\
& s_{ijt}, y_t \in \{0,1\},\ \forall i,j,t \in [1,n]
\end{aligned}
$$

We partially relax the current problem to generate the Lagrangian function as

$$
\begin{aligned}
L(Y,P,S,\lambda) = & a\sum_{t} y_t + b\sum_{i,j,t} P_{ijt} + \sum_{t} \lambda_t^{(Y)}(\sum_{i,j} s_{ijt} - n^2 y_t) \\
& - \sum_{i,j,t} \lambda_{ijt}^{(P0)} P_{ijt} + \sum_{i,j,t} \lambda_{ijt}^{(PM)}(P_{ijt} - P_{max}) \\
& - \sum_{i,j,t} \lambda_{ijt}^{(S)}(G_{ij}P_{ijt} - \beta\sum_{u>i} G_{uj}P_{ujt} - \beta I - \beta N_0 - \Phi(s_{ijt} - 1)) \\
s.t. \quad & \sum_{j,t} s_{ijt} = 1,\ \forall i \in [1, n-1] \\
& \sum_{j,t} s_{njt} = 0 \\
& \sum_{j} s_{ijt} \leq 1 - \frac{1}{n^2}\sum_{l \geq t,j} s_{jil},\ \forall i,t \in [1,n] \\
& s_{jit} + \sum_{u}\sum_{v \in V_i^{(I)}} s_{uvt} \leq 1,\ \forall i,j,u \in [1,n] \\
& \sum_{j \notin V_i^{(T)}} s_{ijt} = 0,\ \forall i \in [1, n-1] \\
& s_{ijt} \in \{0,1\},\ \forall i,j,t \in [1,n]
\end{aligned}
$$

So the KKT conditions can be formulated as follows

$$\sum_{j,t} s_{ijt} = 1,\ \forall i \in [1, n-1]$$

$$\sum_{j,t} s_{njt} = 0$$

$$\sum_{j} s_{ijt} \leq 1 - \frac{1}{n^2} \sum_{l \geq t, j} s_{jil},\ \forall i, t \in [1, n]$$

$$s_{jit} + \sum_{u} \sum_{v \in V_i^{(I)}} s_{uvt} \leq 1,\ \forall i, j, u \in [1, n]$$

$$\sum_{j \notin V_i^{(T)}} s_{ijt} = 0,\ \forall i \in [1, n-1]$$

$$\sum_{i,j} s_{ijt} \leq n^2 y_t,\ \forall t \in [1, n]$$

$$0 \leq P_{ijt} \leq P_{max} \quad \forall t, i, j \in [1, n]$$

$$G_{ij} P_{ijt} - \beta \sum_{u > i} G_{uj} P_{ujt} - \beta I - \beta N_0 \geq \Phi(s_{ijt} - 1) \quad \forall t, i, j \in [1, n]$$

$$\lambda \succeq 0$$

$$\lambda_t^{(Y)} (\sum_{i,j} s_{ijt} - n^2 y_t) = 0,\ \forall t \in [1, n]$$

$$\lambda_{ijt}^{(P0)} P_{ijt} = 0,\ \forall i, j, t \in [1, n]$$

$$\lambda_{ijt}^{(PM)} (P_{ijt} - P_{max}) = 0,\ \forall i, j, t \in [1, n]$$

$$\lambda_{ijt}^{(S)} (G_{ij} P_{ijt} - \beta \sum_{u > i} G_{uj} P_{ujt} - \beta I - \beta N_0 - \Phi(s_{ijt} - 1)) = 0,\ \forall i, j, t \in [1, n]$$

$$a - n^2 \lambda_t^{(Y)} = 0,\ \forall t \in [1, n]$$

$$b - \lambda_{ijt}^{(P0)} + \lambda_{ijt}^{(PM)} - \lambda_{ijt}^{(S)} G_{ij} + \beta \sum_{u < i} \lambda_{ujt}^{(S)} G_{ij} P_{ijt} = 0,\ \forall i, j, t \in [1, n]$$

$$\lambda_t^{(Y)} + \lambda_{ijt}(S) \Phi = 0,\ \forall i, j, t \in [1, n]$$

$$s_{ijt}, y_t \in \{0, 1\},\ \forall i, j, t \in [1, n]$$

I am not clear how to go on with such a long formula yet. My opinion is that our problem may not be solvable by just applying primal-dual decomposition here. Besides, primal-dual decomposition requires the value of dual variables of neighboring nodes and the final result may need a long iteration to update the dual variables. So, the solution may not look like the deterministic aggregation scheduling algorithms with bounded latency bound. Then, I spend the rest of time on designing a distributed aggregation scheduling algorithm.

II. Distributed Algorithm

I have the skeleton of distributed algorithm in previous reports. Here I present more details on the implementation.

A. Tree Construction

Part I: Node Level Labeling

The sink node sends out a packet with value $l = 1$ to each of its neighbors. Once the neighbor of sink receives the packet of value $l = 1$, it updates its level value $l = 1$ and sends out a packet with value $l = 2$.

For each node, its level value is initialized as $l = n$. And suppose the current level value for one node is $l = i$. If that node receives a packet with $l = j < i$, it updates it level value to $l = j$ and sends out a packet with value $l = j + 1$ to its neighbors. Otherwise, that node just ignores the packet.

Part II: Tree Construction:

After the *node-level-labeling* phase, each node gets its level value, which is its hop-distance $l = i$ to the sink, and a set of neighbors with level value $l = i - 1$ or $l = i$.

Lemma 1: For a node v_i with level value $l = i$, the level value of its neighbor can only be in $\{i - 1, i, i + 1\}$.

Proof: Suppose the node v_i with level value $l = i$ has a neighbor v_j with level value $l = j$.

- If $i - 1 > j$, then the node v_i should have received a packet from v_j with value $l = j + 1 < i$. Then the level value of v_i should be less than i. So contradiction occurs.
- If $i + 1 < j$, then the node v_j should have received a packet from v_i with value $l = i + 1$. Then the level value of v_j should be $i + 1$ instead of j. So contradiction occurs.

In conclusion, for a node v_i with level value $l = i$, the level value of its neighbor can only be in $\{i-1, i, i+1\}$. Otherwise, we will have contradiction. ■

Distributed Tree Construction:

For each node v_i with level value l, it searches its neighboring node set with level value $l - 1$ and finds the nearest one v_j in that set. Then we check the current children set for v_j to see whether v_i can be decoded with them by SIC. If so, we establish a directed link e_{ij} from v_i to v_j. Otherwise, we check the next nearest neighbor with level $l - 1$ and repeat the above procedure again. In the end, if one link is established between v_i and one of its neighbor node with level $l - 1$, algorithm ends. Otherwise, v_i will check the neighboring nodes with level l, which has already connected to some node with level $l - 1$. If the set is not empty, v_i connects to the nearest node in that set. Otherwise, v_i connects to the original nearest node in set of level $l - 1$.

We can prove that, in this way, we can finally get an aggregation tree.

Proof: For each node v_i, we only establish one link starting from it. So, in the end, we have a link set of exactly $n - 1$ links for the $n - 1$ non-sink nodes, which means that the resulting graph is a directed acyclic graph with only one node v_n (the sink) with out-degree 0. In another word, the graph is a tree rooted at the sink. ■

B. Link Scheduling

The link scheduling is distributed and each node will run a Finite State Machine on it with following states.

- **S**: Successfully Scheduled. It means the node has already received the data from all its children or it is a leaf node in the tree.
- **NR**: Not Ready to receive. None of its children is in the state of S.
- **R0**: Ready to receive date from part of its children. At least one but not all of its children is in the state of S.
- **R1**: Ready to receive date from all of its children. All of its children are in the state of S.

Distributed Link Scheduling:

Step 1: At the beginning of the link scheduling algorithm, each leaf node is initialized into state S and each internal node is initialized into state NR. Each leaf node will inform its parent node about its state.

Step 2: Each internal node will check the states of its children in the tree. If none of them is in state S, it will remain in state NR. If at least one but not all of its children is in the state of S, it will change into state $R0$ and sends out a packet with it ID number, level value and state value $R0$ ($< ID, l, R0 >$) to all internal nodes, whose state is not S, in a range of r_c (defined as a guard distance in previous report). Otherwise, it will change into state $R1$ and sends out a packet with it ID number, level value and state value $R1$ ($< ID, l, R1 >$) to all internal nodes, whose state is not S, in a range of r_c (defined as a guard distance in previous report).

Step 3:

- Each node in state $R0$ will check the states of possible receivers in a range of r_c. If no other receiver is in state $R1$ and it is the one with largest ID number in the set of receivers with the largest level value and state $R0$, it will be scheduled to receive data from its children and transfer into state S, which is informed to its parent node. Otherwise, it will return to NR.
- Each node in state $R1$ will check the states of possible receivers in a range of r_c. If it is the one with largest ID number in the set of receivers with the largest level value and state $R1$, it will be scheduled to receive data from its children and transfer into state S, which is informed to its parent node. Otherwise, it will return to NR.

Step 4: Repeat Step 2 - 3, until the state of sink is S.

The next work is to analyze the aggregation latency bound and energy consumption. I have some results now. I need more lemmas and theories to complete the story.