Backhaul Design and Controller Placement for Cooperative Mobile Access Networks

Thorsten Biermann*, Luca Scalia†, Jörg Widmer‡, and Holger Karl*

*DOCOMO Euro-Labs, Munich, Germany, †Institute IMDEA Networks, Madrid, Spain, ‡University of Paderborn, Germany

{biermann, Scalia}@docomolab-euro.com, widmer@imdea.org, holger.karl@upb.de

Abstract—Exploiting base station cooperation in wireless mobile access networks leads to benefits in wireless transmission capacity, inter-cell interference management, and cell edge user experience. The clustering of cooperative Base Station (BS) sets, necessary for achieving the desired wireless performance, poses several challenges in the backhaul architecture design. This paper addresses the problem of placing and connecting controller/processing nodes within the backhaul infrastructure, which coordinate and/or process signals of cooperating base stations. We formulated a Mixed Integer Linear Programm (MILP) for this problem and a heuristic algorithm that approximates the optimal solution. While the heuristic’s solution quality is close to the optimum, the runtime and memory requirements are multiple orders of magnitude lower compared to solving the MILP. This advantage allows to use the proposed heuristic either for backhaul/core network pre-planning or for on-the-fly network reconfiguration during ongoing mobile network operation.

I. INTRODUCTION

Cooperation in cellular mobile access networks shows high benefits in terms of wireless transmission capacity, inter-cell interference management, and cell edge user experience. Several techniques have been proposed so far (joint precoding and decoding, inter-cell coordination, etc.), each of them having different merits.

There has been a lot of research in the area of implementing and evaluating the performance of different cooperation techniques, like joint signal processing [1]–[3] and inter-cell coordination [4], [5], from the wireless point of view. The results show how to integrate the cooperation techniques into the wireless systems, demonstrate the possible performance gains, and partly scratch the surface on how to design an overall network architecture that incorporates cooperation techniques. Other work covers (i) the synchronization of the components involved in the cooperation schemes [6], (ii) the number of BSs that need to cooperate to achieve the desired gains [7], (iii) the management of inter-cluster interference [8], e.g., by introducing overlapping cluster configurations, and (iv) the efficient collection of Channel State Information (CSI) information [9].

The common denominator of most cooperation techniques are central controller/processing nodes within the core/backhaul network of the mobile operator. Although the impact of a limited wireline network has been studied [10], the problem of where to optimally place these nodes inside the network and how to connect them to the BSs is not addressed. For this, requirements of the cooperation techniques and properties of the wireline network both have to be taken into account. Otherwise, the wireless performance decreases or cooperation even becomes impossible.

In the following, we first formulate the problem that is addressed in this paper in detail in Sec. II before we present a MILP that solves this problem optimally (Sec. III). Thereafter, we propose a heuristic algorithm in Sec. IV that approximates the optimal solution while reducing the runtime and memory requirements. Sec. V presents an evaluation of both approaches. Finally, we discuss application scenarios of the heuristic (Sec. VI) before concluding our work in Sec. VII.

II. PROBLEM DESCRIPTION

Clustering cooperative BS sets, necessary for achieving the desired wireless performance, poses several challenges in the backhaul architecture design. Depending on the used cooperation scheme, a central controller and/or processing node and a set of cooperating BSs need to exchange signaling and sometimes user data traffic.

The placement of these controller/processing nodes and their connection to the cooperating BSs play a crucial role in the overall backhaul architecture design. System parameters like propagation delay in the wired part of the network, required link capacity for transporting user data and control information, synchronization between cooperative BSs, Capital Expense (CAPEX) and Operating Expense (OPEX) of the backhaul architecture need to be taken into account.

This leads to the following problem for designing backhaul networks for cooperative, cellular mobile access networks. Given a set of BSs and their (potential) interconnections, we need to know how to optimally position controller/processing nodes in the network and how to assign BSs to these nodes.

To solve this network design problem, we use the parameters in Tab. I to define the input scenario. These parameters define a graph that has a vertex for each BS $b \in B$ and an edge for each link $l \in L$ between the BSs. All vertices and edges are augmented with properties, like capacity and latency, that are important for the network design process. Additionally, there are four global properties ($C_{\text{min}}$, $C_{\text{max}}$, $t_{\text{proc}}$, and $t_{\text{max}}$) that are independent of BSs or links.

III. OPTIMAL SOLUTION

This section introduces a Mixed Integer Linear Programm (MILP) that optimally solves the network design problem.
introduced in Sec. II. Based on the input, the MILP calculates optimal positions for the controller/processing nodes and assigns BSs to these nodes while minimizing costs. This assignment leads to the optimal interconnection of all BSs.

In the following, we will use monetary costs as optimization goal. This can easily be adapted to a different cost metric like energy consumption.

The MILP uses the parameters in Tab. I as input and uses the variables shown in Tab. II to optimize the metric.

**TABLE I: Input scenario parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B$</td>
<td>Set of BSs</td>
</tr>
<tr>
<td>$L$</td>
<td>Set of links, where $L \subseteq B \times B$</td>
</tr>
<tr>
<td>$b_{\text{costF}}$</td>
<td>Fixed costs for setting up a controller at BS $u \in B$</td>
</tr>
<tr>
<td>$b_{\text{costC}}$</td>
<td>Costs per controlled BS for a controller at BS $u \in B$</td>
</tr>
<tr>
<td>$b_{\text{cap}}$</td>
<td>Required capacity (e.g., bandwidth/data rate) at BS $u \in B$</td>
</tr>
<tr>
<td>$l_{\text{cap},u,v}$</td>
<td>Capacity (e.g., bandwidth/data rate) of link $(u,v) \in L$</td>
</tr>
<tr>
<td>$l_{\text{f},u,v}$</td>
<td>Latency of link $(u,v) \in L$</td>
</tr>
<tr>
<td>$l_{\text{cost},u,v}$</td>
<td>Costs of link $(u,v) \in L$</td>
</tr>
<tr>
<td>$c_{\text{min}}$</td>
<td>Minimum required cluster size</td>
</tr>
<tr>
<td>$c_{\text{max}}$</td>
<td>Maximum allowed cluster size</td>
</tr>
<tr>
<td>$l_{\text{proc}}$</td>
<td>Processing time at controller</td>
</tr>
<tr>
<td>$l_{\text{max}}$</td>
<td>Maximum total round trip time from BS to controller</td>
</tr>
</tbody>
</table>

**TABLE II: Optimization variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_{\text{act},u,v}$</td>
<td>Determines whether link $(u,v) \in L$ is active, $l_{\text{act},u,v} \in {0,1}$</td>
</tr>
<tr>
<td>$b_{\text{act}}$</td>
<td>Determines whether a controller/processing node is colocated at BS $u \in B$, $b_{\text{act},u} \in {0,1}$</td>
</tr>
<tr>
<td>$f_{s,d,u,v}$</td>
<td>Determines whether a data flow from a controller/processing node at $s \in B$ goes to $d \in B$ over link $(u,v) \in L$, $f_{s,d,u,v} \in {0,1}$</td>
</tr>
<tr>
<td>$b_{\text{actCostF}}$</td>
<td>Actual fixed controller/processor costs at $u \in B$, $b_{\text{actCostF}}u \geq 0$</td>
</tr>
<tr>
<td>$b_{\text{actCostC}}$</td>
<td>Actual controller/processor costs at $u \in B$ per each controlled BS, $b_{\text{actCostC}}u \geq 0$</td>
</tr>
<tr>
<td>$l_{\text{actCost},u,v}$</td>
<td>Actual costs for link $(u,v) \in L$, $l_{\text{actCost},u,v} \geq 0$</td>
</tr>
</tbody>
</table>

Setting $l_{\text{act},u,v}$ to 1 means that the link $(u,v)$ transports some flow, i.e., there is at least one $f_{s,d,u,v}$ set to 1. Similar to this, if $b_{\text{act}}$ is set to 1, a controller/processing function is colocated at BS $u$. At the moment, our MILP supports to colocate these functions at a BS; they cannot be positioned arbitrarily in the network.

The three variables $b_{\text{actCostF}}u$, $b_{\text{actCostC}}u$, and $l_{\text{actCost},u,v}$ contain the actual costs for controllers and links. These costs depend on whether a controller/processing node is actually colocated at a BS and on whether a link is active, i.e., used to connect clustered BSs, or not.

To get the total link and controller/processing node costs for a network configuration, we need to sum up the individual costs. This is done in Eq. (1) and Eq. (2), respectively.

$$l_{\text{cost}} = \sum_{u \in B} l_{\text{actCost},u,v}$$

$$b_{\text{cost}} = \sum_{u \in B} b_{\text{actCostF}}u + b_{\text{actCostC}}u$$

The goal is to minimize the total costs while taking into account the constraints of the applied wireless cooperation scheme, defined by the parameters in Tab. I. The following MILP achieves that:

$$\min \quad b_{\text{cost}} + l_{\text{cost}}$$

s.t.

$$\forall s \in B, (u,d) \in L$$

$$\sum_{s \in B, (u,d) \in L} f_{s,d,u,d} = 1, \quad \forall d \in B,$$

$$\forall s \in B, (u,d) \in L$$

$$\sum_{s \in B, (u,d) \in L} f_{s,d,u,s} = 1, \quad \forall d \in B,$$

$$\forall u \in B, (s,d) \in B \times B, u \neq s, u \neq d,$$

$$f_{s,d,u,v} = 0, \forall (s,d,u,v) \in B \times B \times B \times B,$$

$$s \neq d, u = v,$$

$$1_{\text{act},u,v} \cdot M \geq \sum_{(s,d) \in B \times B} f_{s,d,u,v}, \forall (u,v) \in L,$$

$$\forall c \in B,$$

$$\sum_{(d,v) \in B \times B, (c,v) \in L} f_{c,d,c,v} \geq c_{\text{min}} \cdot b_{\text{act}},$$

$$\forall c \in B,$$

$$\sum_{(d,v) \in B \times B, (c,v) \in L} f_{c,d,c,v} \leq c_{\text{max}}, \forall c \in B,$$

$$\forall (u,v) \in L,$$

$$f_{s,d,u,v} \cdot b_{\text{cap}}d \leq 1_{\text{cap}},$$

$$\forall (u,v) \in L,$$

$$\forall (u,v) \in L,$$

$$\sum_{(u,v) \in L} f_{s,d,u,v} \cdot (l_{\text{u},u,v} + l_{\text{v},u}) \leq l_{\text{max}} - l_{\text{proc}},$$

$$\forall (s,d) \in B \times B,$$

$$b_{\text{actCostF}}c = b_{\text{act}}, \quad b_{\text{actCostC}}c, \forall c \in B,$$

$$b_{\text{actCostC}}c = \sum_{d \in B, (c,v) \in L} f_{c,d,c,v} \cdot b_{\text{actCostC}}c,$$

$$\forall c \in B,$$

$$l_{\text{actCost},u,v} = l_{\text{act},u,v} \cdot l_{\text{cost},u,v}, \forall (u,v) \in L,$$
The first constraint in Eq. (4) ensures that each BS is assigned to exactly one controller. This is done by fixing the number of flows that end at each BS to 1. The second constraint, shown in Eq. (5), guarantees that each flow that ends at a BS also has a start BS – the controller. Eq. (6) contains the third constraint, which is necessary to create the flows in the network. It guarantees the flow balance, i.e., whenever a flow enters a node it also has to leave it again, except the node is the flow’s source or destination. Finally, Eq. (7) forbids local loops for each flow.

After having created the flows between the controllers and the BSs, the helper constraint in Eq. (8) activates all links in the network that are required to transport one of the flows. This constraint uses a “big-M constant” to achieve this. Similarly, the BSs, the helper constraint in Eq. (8) activates all links in the network that are required to transport one of the flows. This might be useful in case the controller capacities are limited.

As many cooperation schemes, like joint precoding, require a minimum amount of BSs being jointly controlled to achieve the desired gains, the constraint in Eq. (10) requires at least $c_{\text{max}}$ BSs to be connected to the same controller. At the same time, Eq. (11) limits the maximum cluster size to $c_{\text{max}}$. This might be useful in case the controller capacities are limited and hence clusters cannot exceed a certain size.

The constraints in Eq. (12) and Eq. (13) ensure that the link capacities are not exceeded in the downlink (from controller to BS) and in the uplink, respectively.

The next constraint in Eq. (14) ensures that the round-trip delay between a controller and a BS is below an upper bound. This bound is calculated by subtracting the required processing time $t_{\text{proc}}$ at the controller from the maximum allowed delay $t_{\text{max}}$. Such a constraint is important, e.g., for joint precoding. Here, the encoded data must be sent while the CSI, based on which the encoding was done, is still valid.

Finally, the last three constraints calculate the link and controller costs for the resulting network. Eq. (15) and Eq. (16) define the fixed and dynamic controller costs. Eq. (17) sets the costs for each link depending on whether it is active or not.

The presented MILP returns the optimal solution for the problem in Sec. II. Finding such a solution, however, is NP-hard and requires a long runtime solver.

**IV. HEURISTIC APPROACH**

To overcome the MILP’s runtime problem, we propose a heuristic to approximate the optimal solution. The heuristic takes the same input as the MILP (i.e., a property graph containing the BSs and potential interconnections) and produces the same kind of output (locations of controllers/processors and optimized interconnections between these components).

We have chosen a greedy approach based on a modified Breadth First Search (BFS) algorithm to build clusters of BSs. The resulting heuristic is summarized in Alg. 1.

The method configureCoopNetwork expects the input parameter $G_{\text{in}}$. This is the input graph of BSs and potential interconnections. The algorithm first initializes the output graph $G_{\text{out}}$. After this, it traverses a loop where in each iteration one new cluster is created and added to $G_{\text{out}}$. Furthermore, BSs that have been assigned to a cluster are removed from $G_{\text{in}}$.

At the beginning of each iteration, one of the BSs in $G_{\text{in}}$ is selected that will act as a cluster controller. Starting from this new controller $s$, the BFS’ algorithm builds the cluster around $s$ and returns the resulting cluster graph $G_{\text{newClust}}$. This graph contains all BSs that are member of the cluster and links that connect these BSs to the controller $s$.

The way how to choose $s$ from $G_{\text{in}}$ plays an important role for the quality of the heuristic’s output. The method chooseController selects a BS from $G_{\text{in}}$ such that its distance to the border of $G_{\text{in}}$ equals the expected radius of the cluster that is currently created.

The size (and hence the radius) of the resulting cluster depends on different things: the physical properties of the involved BSs and their interconnections, and the constraints that are imposed by the wireless cooperation technique.

In the following, we give an example how this can be done for joint precoding as cooperation technique (Alg. 2).

**Algorithm 1 configureCoopNetwork($G_{\text{in}}(B, L)$)**

1: $G_{\text{out}} = \emptyset$ // initialize output graph with empty set
2: while $|B| > 0$ do
3: $s = \text{chooseController}(G_{\text{in}})$
4: $G_{\text{newClust}} = \text{BFS}'(G_{\text{in}}, s)$ // calculate new cluster
5: $G_{\text{out}} = G_{\text{out}} \cup G_{\text{newClust}}$ // add new cluster to $G_{\text{out}}$
6: $G_{\text{in}} = G_{\text{in}} \setminus G_{\text{newClust}}$ // remove new cluster from $G_{\text{in}}$
7: end while
8: return $G_{\text{out}}$

Joint precoding has two major constraints that have to be fulfilled. The first one is that the latency from measuring the CSI until the point in time where the jointly encoded data is sent must be below an upper bound, e.g., 1 ms for Long Term Evolution (LTE). Hence, the processing delay at the controller and the Round Trip Time (RTT) from the controller to the controlled BSs must not exceed this limit. Accordingly, the expected cluster radius considering the latency $c_{\text{lat}}$ can be calculated as shown in the first line of Alg. 2. The parameters $t_{\text{max}}$ and $t_{\text{proc}}$ contain the maximum allowed time span from measuring the CSI to the actual sending of the data, and the processing delay at the controller, respectively (as defined in Tab. 1). Their difference is divided by the mean link latency.

The second limitation for joint precoding is that the links must have enough capacity to transport the signals from the controller to the BSs from where they are sent. So the amount of bandwidth that is required at each BS and the link capacities from the controller to the BSs (possibly involving multiple hops) limit the maximum possible cluster size, too. This is taken into account in line 2 of Alg. 2. There, the expected cluster radius depending on the capacity properties of the input
The input scenario consists of a matrix of \( n \) BSs that form a square. We look at two different arrangements of the BSs. The first is a regular arrangement where all BSs have identical inter-BS distances of \( \pi = 1600 \sqrt{3} \text{m} \approx 2771 \text{m} \) [11] and have potential interconnections to all neighboring BSs. The second type of arrangement is irregular and adds randomness to the BS positioning. Hereby, BSs still have a mean distance of \( \pi \) but positions are normally-distributed with standard deviation \( \pi/8 \). BSs are interconnected if their distance is below \( 1.25 \cdot \pi \).

The reason for having irregular input scenarios is to check whether our heuristic suffers from arbitrarily choosing new neighbors during the BFS. Fig. 1 shows two examples of these two arrangement types for \( n = 16 \).

The interconnections are optical and have a latency of \( s \cdot \frac{45}{c} \), where \( c \) is the speed of light and 1.45 the refraction index of the fiber. This corresponds to a typical Single-Mode Fiber (SMF) setup. We further assume a Radio over Fiber (RoF) setup from the controllers to the BSs.\(^1\) The RoF capacity of each link is 3000 MHz, which corresponds to standard RoF equipment. Every BSs consists of 3 sectors that have 4 antennas each, operated at a bandwidth of 100 MHz, which

\(^1\)Note that this only affects the evaluation. Our proposed methods are generic enough to support any technology.
In the offline case, the heuristic approach is applied before deploying the actual backhaul/core network infrastructure. This corresponds to a network planning procedure when a mobile operator intends to augment its existing network with base station cooperation.

In the online case, the cooperative network is already deployed. Here, it can be required to, e.g., re-assign certain base stations to other cooperation clusters to react on changes in the network, like link failures or varying load. This information about the network is collected during its operation and provided as input data to the heuristic, which determines what to change in the network to improve the operation.

### VII. Conclusions

We proposed two methods to solve the problem of how to position controller/processor nodes in a cooperative mobile access network and how to assign BSs to them. The MILP returns optimal solutions but is only feasible for small input scenarios. The heuristic produces solutions that are only 5% worse than the MILP's output but has a runtime and memory consumption that is magnitudes lower compared to the MILP.

The heuristic allows to calculate quasi-optimal solutions for large input scenarios and can be applied during the runtime of a network to reconfigure it on the fly, thanks to the low resource requirements. Next steps are to extend the heuristic to position controllers independently of the BS locations, to take into account shared resources in tree or ring topologies, and to support multiple overlapping clusters per BS.

### REFERENCES


