Cloud-based Spectrum Sharing in Virtual Wireless Networks

Fatemeh Shirzad
Department of Computer Science
University of Calgary
Calgary, Canada
Email: fatemeh.shirzad@ucalgary.ca

Majid Ghaderi
Departmnet of Computer Science
University of Calgary
Calgary, Canada
Email: mghaderi@cs.ucalgary.ca

***Abstract*—This paper studies the network-wide spectrum sharing problem in virtual cellular networks applicable to Cloud Radio Access Network architecture. Virtual wireless networks share network resources such as time-frequency resources on the same physical infrastructure. A critical problem is then the allocation of shared physical resources to the virtual networks so that the utilization of the resources is maximized. We formulate the problem as a linear integer maximization problem, which is shown to be NP-hard. We then develop a polynomial time heuristic algorithm called Non-balancing Spectrum Sharing (NSS), which is guaranteed to achieve a solution whose resource utilization approaches half of that of the optimal solution in the worst-case. Two additional heuristic algorithms are also proposed to improve the worst-case performance of NSS by enabling load balancing among adjacent base stations. We have simulated the proposed algorithms and the optimal algorithm under different network configurations. The simulation results confirm that i) NSS performs remarkably close to the optimal algorithm, and ii) the two other heuristic algorithms outperform NSS, and consequently are even closer to the optimal algorithm in the simulated scenarios.**

***Index Terms*—Virtualization; Spectrum Sharing; Wireless Network Embedding; Virtual Resource Allocation; Cloud Radio Access Networks; Cellular Networks;**

I. Introduction

A. Motivation

One of the problems facing cellular networks today is the scarcity of wireless resources because of an upsurge in mobile data traffic. According to [1], mobile data will experience tenfold growth from 2014 to 2019. One approach to address this problem is to add new infrastructure such as new base stations to improve spatial frequency reuse. However, this approach would increase both the fixed and operational costs of the network, while the revenue of the network operator would not increase at the same pace because of government regulations and competition in the market.

While current cellular networks are faced with a shortage of radio resources, the radio frequencies currently allocated to these networks are actually underutilized [2]. Extensive research is being conducted to address this problem and design networks that are more efficient in utilizing limited radio resources [3].

One of the promising proposals to address the underutilization of radio resources in cellular networks is based on radio access network (RAN) *virtualization*. RAN virtualization refers to the sharing of physical radio network resources among several *virtual networks*. Network virtualization, in general, improves resource utilization for physical network owners and reduces management and equipment costs for virtual network operators [3].

With RAN virtualization, several Mobile Virtual Network Operators (MVNOs) lease infrastructure and spectrum from Mobile Network Operators (MNOs), who own the network infrastructure and spectrum. This reduces start-up costs and time for MVNOs. Today, MVNOs lease spectrum statically, which means MVNOs acquire sufficient spectrum to support peak traffic demand. However, there are times in which the actual demand is much lower than the peak demand. This results in an underutilization of resources. RAN virtualization enables dynamic allocation of spectrum resources to MVNOs, which results in higher utilization of radio and infrastructure resources.

One of the main issues in realizing RAN virtualization is resource sharing, particularly spectrum sharing, among multiple virtual RANs. We note that several phrases are used in the literature to refer to this problem, such as spectrum sharing, wireless resource allocation, and wireless virtual network embedding. We will use these phrases interchangeably throughout the paper when appropriate.

A few solutions have recently been proposed in the literature to address the resource sharing problem in virtual RANs [4]–[10], which will be discussed later in Subsection I-C. The problem with these existing solutions is that they consider base stations *independently*. However, in cellular networks, resource allocation on one base station may affect resource allocation on other base stations. In fact, we can accommodate requests in regions that are common to more than one base station in such a way that more balanced allocation is achieved, *i.e.*, one base station is not overwhelmed with requests. By distributing the load more evenly among neighboring base stations, fewer resource requests from virtual networks are rejected. Thus, more resource requests are accommodated across the base stations, which leads to a higher utilization of radio resources.

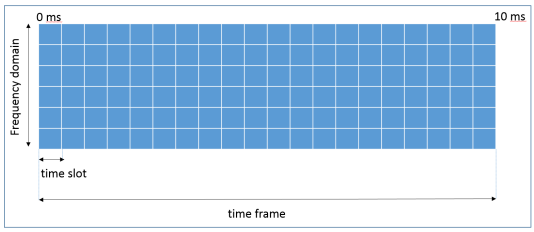


Fig. 1: The structure of a *frame* in LTE cellular systems.

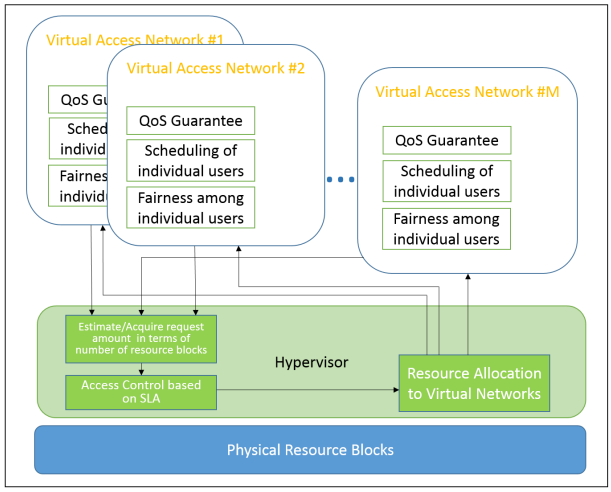


Fig. 2: The conceptual architecture of the virtualized RAN.

B. Our Work

In this paper, we investigate the spectrum sharing problem from a radio access network point of view. In fact, we take a network-wide view towards resource sharing as opposed to treating base stations individually as with the above-mentioned works. In our approach, resource allocation decisions are made in a relatively *short time scale*, which is in the order of user scheduling operations. Considering short time scale is in accordance with the recommendation made by Third Generation Partnership Project (3GPP) about network sharing in future networks [11]. For example, in Long-Term Evolution (LTE) networks, user scheduling decisions are made at the beginning of each scheduling *frame*, where a frame consists of 10 timeslots and each timeslot is 1 ms (see Fig. 1). Making frame-level decisions implies that a virtual network knows exactly the amount of its resource requirement over each frame and hence can ask the network for that much requirement at the beginning of the frame.

We focus on Cloud-RAN architecture [12], in which spectrum sharing can be performed in a centralized way. Therefore, we assume that a central *hypervisor* controls sharing of spectrum to virtual networks (see Fig. 2). The objective of the spectrum sharing algorithm on the hypervisor is to accommodate as many resource requests as possible.

Our main contributions, in this paper, are summarized as follows.

- We formulate the spectrum sharing problem in the cellular network virtualization context as a linear integer maximization problem, which is shown to be NP-hard.
- A $\frac{1}{2}$-approximation algorithm of polynomial time complexity called Non-balancing Spectrum Sharing (NSS) is developed to solve the problem. We analyze the computational complexity of NSS and prove that it achieves the approximation ratio $\frac{1}{2}$.
- Two other heuristic solutions called Balancing Spectrum Sharing 1 (BSS1) and Balancing Spectrum Sharing 2 (BSS2) are proposed based on the idea of balancing the load among neighboring base stations. Using simulations, we show that BSS1 and BSS2 have better worst-case performances than NSS.
- We study the performances of the three proposed algorithms and the optimal algorithm via simulation.

C. Related Work

Network virtualization in wired networks has been investigated extensively [13]. However, wireless network virtualization is still in its infancy and faces several challenges that need to be addressed in order to have a practical solution [3]. One of the main issues, in this regard, is resource sharing, particularly spectrum sharing among wireless virtual networks.

A framework in [4] is proposed for the spectrum sharing problem. In this framework, a physical network has different dimensions such as frequency, time, and space. Each virtual request is a set of resource blocks with the same dimensions as those of the physical network, and the problem is to place these blocks on the physical network so that maximum revenue is achieved. Based on this framework, a number of heuristic solutions are proposed in [5] and [6]. Another approach to the spectrum sharing problem is presented in [10], where the authors introduce Network Virtualization Substrate (NVS) with two levels of scheduling: slice level (virtual network level) and flow level. Other solutions for spectrum sharing in wireless networks are presented in [7]–[9].

Perhaps the closest work to ours is the work presented in [14]. Although [14] presents a network-wide approach to spectrum sharing, it does not consider resource requests over a short time scale because of signaling overheads. This restriction is primarily due to the distributed network architecture they have considered which is the traditional LTE-type cellular architecture. Our approach is based on allocating virtual network resource requests over a time frame. This means that the problem is formulated and solved at the beginning of each time frame. The importance of working at such a short time scale is that virtual networks require such capability in order to have control over their user scheduling decisions. Moreover, it is more feasible for virtual networks to have more accurate information about their resource demands [5].

D. Paper Organization

The rest of the paper is organized as follows. In Section II, we describe our system model and present a formulation of the problem as a linear integer maximization problem. Our proposed algorithms and their analysis are described in

Section III. The simulation results are presented in Section IV. Section V concludes the paper.

II. System Model

In this section, we describe the system architecture, the resource model, and the problem formulation.

A. System Architecture

Different components of the system considered in this paper are depicted in Fig. 2. As shown in the figure, the task of scheduling individual users is the responsibility of each virtual network. The virtualization layer, called *hypervisor*, allocates physical resource blocks to virtual networks based on the resource requests it receives from them. In this architecture, virtual networks have full control and visibility over the allocation of their resources to their users. For instance, a virtual network may wish to consider Quality of Service (QoS) requirements and/or fairness among individual users, while another virtual network only considers revenue maximization when scheduling individual users. This customization property is a key requirement of a virtualization system [3].

There are three units in the hypervisor layer. One unit estimates or acquires the required amount of resources. The second unit, access control unit, ensures that Service Level Agreements (SLAs) are satisfied by providing fairness and isolation among virtual networks. If SLAs are satisfied, the resource requests are then sent to the third unit, resource allocation unit. The resource allocation unit runs the spectrum sharing algorithm, which could be one of the algorithms proposed in this paper.

Our proposed algorithms are designed for Cloud-RAN architecture, in which the functionality of a traditional base station is divided between remote radio heads (RRHs) and baseband processing units (BBU) hosted in a center called BBU pool. RRH and BBU perform the radio interface and processing functionalities, respectively. BBUs are pooled together in the data center and are shared among base stations. Fig. 3 shows the high-level architecture of a Cloud-RAN. We utilize this architecture to develop a network-level resource allocation mechanism. Interested readers may refer to [12] for more details on Cloud-RAN.

B. Resource Model

We assume that the physical resources are divided into a number of orthogonal time-frequency resources such as in LTE systems. Each of these resources is called a *resource block*, which is the smallest unit that can be allocated to a user. Following the LTE architecture, a set of resource blocks in time and frequency domain is shown in Fig. 1. A *time slot* is the duration of one resource block in time. A number of time slots form a *time frame*. The resource allocation problem is solved at the beginning of each time frame for downlink traffic. The available resources for each RRH are a subset of the total resource blocks in a frame. As provisioned in LTE systems, we assume that interference between any two RRHs is coordinated using techniques such as Inter-Cell Interference Coordination (ICIC) and its various extensions [15]. By this separation of interference management and virtual network embedding, we can develop a formulation that is a variant of the *Knapsack* problem [16]. This way, the problem has less integer variables and constraints; therefore, small to medium size problems can be solved optimally using off-the-shelf optimization algorithms.

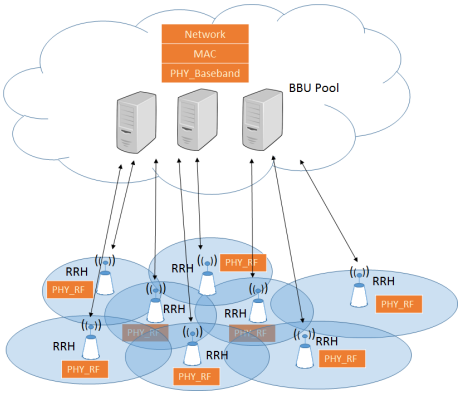


Fig. 3: CRAN architecture and its components.

C. Notation

The list of notations used throughout the paper is presented in Table I. Below, we explain some of the notations in more detail:

- Each virtual network, in each time frame, has an associated resource request called a virtual request. Every virtual request is composed of a number of virtual *sub-requests*. The set $\{vsr_1, vsr_2, ..., vsr_N\}$ represents the sub-requests that are to be scheduled in one frame.
- Each virtual sub-request contains a number of requested resource blocks in a specific geographical region for a group of individual users. That is, for all j, vsr_j is a tuple $\langle \mathcal{B}_j, w_j, t_j \rangle$, in which $\mathcal{B}_j$ is a set of RRH indices to which vsr_j is assignable. For instance, let $\mathcal{B}_j$ be $\{1, 2, 4\}$; it means that any of rrh_1, rrh_2, and rrh_4 can be chosen to serve vsr_j. This happens when vsr_j is in the coverage region of these three RRHs.
- The waiting time for vsr_j is t_j, *i.e.*, vsr_j must be accommodated within t_j time frames after entering the system, otherwise it leaves the system when waiting time is over.
- For any rrh_i, the set $\mathcal{U}_i$ contains the index of any vsr_j if i is in $\mathcal{B}_j$. It is, in fact, composed of indices of all sub-requests that are assignable to rrh_i.

D. Problem Formulation

The objective of the spectrum sharing problem is to maximize the utilization of the system, *i.e.*, maximizing the total number of resource blocks allocated to virtual networks.

TABLE I: Table of notations used in the paper.

Notation	Meaning
B	Number of RRHs
rrh_i	RRH i
c_i	Capacity of rrh_i in terms of available resource blocks
N	Total number of sub-requests
vsr_j	Virtual sub-request j
$\mathcal{B}_j$	Set of indices of all RRHs to which vsr_j is assignable
$\mathcal{U}_i$	Set of indices of all vsr_j that i is in $\mathcal{B}_j$
t_j	Waiting time of vsr_j
w_j	Number of requested resource blocks in sub-request vsr_j

The problem is formulated as the following linear integer optimization problem in each time frame:

$$\textit{maximize} \quad \sum_{i=1}^{B} \sum_{j \in \mathcal{U}_i} w_j \cdot x_{ij} \tag{1}$$

$$\textit{subject to} \quad \sum_{j \in \mathcal{U}_i} w_j \cdot x_{ij} \leq c_i, \quad 1 \leq i \leq B \tag{2}$$

$$\sum_{i \in \mathcal{B}_j} x_{ij} \leq 1, \quad 1 \leq j \leq N \tag{3}$$

$$x_{ij} \ \in \ \{0,1\}, \quad 1 \leq i \leq B, \ 1 \leq j \leq N \tag{4}$$

In this formulation, x_{ij} is an integer variable defined as follows:

$$x_{ij} = \begin{cases} 1 & \text{if } rrh_i \text{ serves } vsr_j \\ 0 & \text{otherwise} \end{cases}$$

The constraint (2) enforces resource limitation of each RRH, *i.e.*, the total number of resource blocks allocated to the sub-requests in rrh_i is at most equal to the capacity c_i. The constraint (3) states that each sub-request vsr_j must be accommodated by at most one RRH in $\mathcal{B}_j$.

III. Spectrum Sharing Algorithms

A. Complexity of the Problem

The problem, as formulated in the previous section, is a variant of the Knapsack problem called Multiple Knapsack with Assignment Restriction (MKAR). It is shown that MKAR is NP-hard [16]. Therefore, approximation algorithms are the only choice to solve it at the present time. Although the Knapsack problem has been studied extensively, to the best of our knowledge, the only result on MKAR is presented in [17].

The solution presented in [17] is based on successively solving single Knapsack problems. In this approach, knapsacks are chosen in an arbitrary order. Once a knapsack is chosen then a single Knapsack problem is solved with the items assignable to it. The accommodated items are subsequently eliminated from the item list of the remaining knapsacks. The algorithm used to solve each single Knapsack problem determines the efficiency of the main algorithm. In the best case, if an optimal single knapsack algorithm is used, the approximation ratio is $\frac{1}{2}$. This means that the solution value in the worst-case is half of the optimal solution value. This ratio is only achieved by optimally solving the single Knapsack problem, which has exponential running time because the Knapsack problem is NP-hard [16].

In this section, we present three algorithms to solve the problem. The first algorithm called Non-balancing Spectrum Sharing (NSS) is designed based on the framework of [17]. We show that for our specific spectrum sharing problem, under some natural assumptions that are often true in cellular networks, we can approach the approximation ratio of $\frac{1}{2}$ without any need to solve the single Knapsack problem optimally. Thus, NSS runs in polynomial time. We also demonstrate, by an example, that the ratio $\frac{1}{2}$ is tight. Next, we develop two other heuristic algorithms called Balancing Spectrum Sharing 1 (BSS1) and Balancing Spectrum Sharing 2 (BSS2), which balance the load among nearby base stations. We conjecture that their approximation ratios are better than that of NSS, but we can currently show this only with simulations.

B. Description of NSS

NSS is summarized in Algorithm 1. It considers the RRHs sequentially in an arbitrary order. It tries to accommodate the assignable sub-requests to each considered RRH. For each rrh_i, the assignable sub-requests (sub-requests in $\mathcal{U}_i$) are sorted in a non-increasing order in terms of w_js. Then, each sub-request vsr_j, in this order, is examined to see if w_j exceeds the current capacity of rrh_i. If it does not exceed, the RRH's current capacity is decreased by w_j. This means that vsr_j is assigned to rrh_i in the current time frame. All assigned sub-requests to rrh_i are removed from the list of assignable sub-requests to the remaining RRHs.

Algorithm 1 Non-balancing Spectrum Sharing (NSS)

```
for each i ∈ {1, ..., B} do
    Sort the sub-requests in U_i in non-increasing order in terms of their requested weights (w_j s)
end for
for each i ∈ {1, ..., B} do
    for each j ∈ sorted U_i do
        if w_j ≤ c_i then
            Assign vsr_j to rrh_i
            c_i = c_i − w_j
            Remove vsr_j from U of all RRHs in B_j while keeping them sorted
        end if
    end for
end for
```

C. Analysis of NSS

We prove that with an upper bound on the size of each sub-request, NSS achieves a $\frac{1}{2}$-approximation ratio in polynomial time. In particular, we assume that $w_j \leq q \cdot c_i$ for any rrh_i and any vsr_j assignable to it, for some $0 < q < 1$. This assumption on the size of sub-requests is reasonable in the cellular network context as the probability that one virtual network requests a large fraction of resources for only one time frame is low. Even if this happens, it is probable that the sub-requests can be divided into several smaller sub-requests. For instance, if a virtual network requests a large portion of available resources

for its users, it is possible to divide this request into several sub-requests with fewer users. To be accurate, the assumption is that for some $0 < q < 1$, the following inequalities hold.

$$w_j \leq q \cdot c_i \ , \qquad \forall j \in \{1, ..., N\} \ , \forall i \in \mathcal{B}_j \tag{5}$$

Using this assumption, the following theorem is proved.

Theorem 1. *NSS achieves a $\frac{1-q}{2-q}$-approximation ratio if the size of any vsr_j is at most $q \cdot c_i$ for some $0 < q < 1$, and for any rrh_i that vsr_j is assignable to.*

To prove Theorem 1, we describe and prove a lemma. First, some definitions are in order. In [17], the *half-full* property is defined as a property of some algorithms for MKAR problem. According to this definition, an algorithm for MKAR is half-full if for every unassigned item at the end of the execution of the algorithm, any knapsack i that the item is assignable to has at least half of its capacity filled.

Here we provide a definition which is a generalization of the half-full property.

Definition 1. An algorithm for the MKAR problem is β-full for some $0 < \beta < 1$ if for every unassigned item at the end of the execution of the algorithm, any knapsack i that the item is assignable to has at least $\beta \cdot c_i$ of its capacity filled.

Now, we prove the following lemma and use it to prove Theorem 1.

Lemma 1. *Any algorithm with β-full property is $\frac{\beta}{\beta+1}$-approximation solution for MKAR problem.*

Proof. Take A as any solution with β-full property with $\mathcal{UI}$ as the set of all unassigned items at the end of its execution. Then, $W(\mathcal{UI})$ denotes the sum of weights of all items in $\mathcal{UI}$. Also, let W_{opt} and W_A denote the solution values of the optimal algorithm and algorithm A, respectively. As A has the β-full property, it follows that:

$$\beta \cdot C(\mathcal{UK}) \leq W(\mathcal{AU}), \tag{6}$$

where $\mathcal{UK}$ denotes the set of all knapsacks with at least one unassigned item among assignable items related to them, and $\mathcal{AU}$ denotes the set of items assigned to knapsacks in $\mathcal{UK}$ in algorithm A. The sum weight of all items in $\mathcal{AU}$ is denoted by $W(\mathcal{AU})$. Let $C(\mathcal{UK})$ denote the sum capacity of all knapsacks in $\mathcal{UK}$. As $\mathcal{AU}$ is a subset of all assigned items, it follows that,

$$W(\mathcal{AU}) \leq W_A \, . \tag{7}$$

From (6) and (7), the following is concluded,

$$\beta \cdot C(\mathcal{UK}) \leq W_A \, . \tag{8}$$

On the other hand, the following inequality holds,

$$W_{opt} \leq C(\mathcal{UK}) + W_A \, . \tag{9}$$

Combining (8) and (9) yields the following results,

$$W_{opt} \leq (1 + \frac{1}{\beta}) \cdot W_A \, .$$

This completes the proof. □

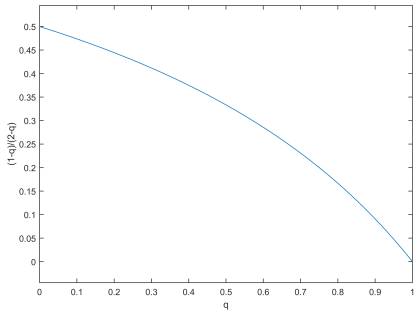


Fig. 4: NSS approximation ratio $(1 - q)/(2 - q)$.

Next, using Lemma 1, we prove Theorem 1.

Proof. The problem is analogous to MKAR problem; sub-requests and RRHs are analogous to items and knapsacks respectively. Therefore, it is sufficient to prove the theorem for MKAR problem. We prove that NSS is $(1 - q)$-full using contradiction as follows. Suppose the algorithm is not $(1-q)$-full. Then, there exists an instance of the problem in which there is a knapsack i which is not $(1 - q)$-full and has an unassigned item j. Then the remaining capacity of knapsack i is more than $q \cdot c_i$. However this is not possible, otherwise item j would have been accommodated as $w_j \leq q \cdot c_i$. Consequently, the algorithm has $(1 - q)$-full property, and using Lemma 1, Theorem 1 is proved. □

According to Theorem 1, the approximation ratio of NSS approaches $\frac{1}{2}$ as q decreases. Fig. 4 depicts the relation between q and the approximation ratio $\frac{1-q}{2-q}$.

Tightness of the bound: We describe an example to show the tightness of the bound. Suppose $q = \frac{1}{10}$; there are two RRHs, rrh_1 and rrh_2 each with M units of resources, and there are 20 sub-requests. Each of the first ten sub-requests needs $\frac{M}{10}$ units of resources, and each of the remaining sub-requests needs $\frac{M}{10} - \epsilon$ units of resources for some small $\epsilon > 0$ arbitrarily close to zero. Assume that the first ten sub-requests can be assigned to both RRHs. Also assume that the remaining sub-requests can only be assigned to rrh_1. In this case, if the algorithm starts with rrh_1, the first ten sub-requests will be chosen to be assigned to rrh_1. When the algorithm proceeds to assign the remaining sub-requests to rrh_2, it finds no assignable sub-requests. The value of this solution is $10 \cdot \frac{M}{10} = M$. However, the optimal solution accommodates the first ten sub-requests on rrh_2 and the remaining sub-requests on rrh_1. The value of the optimal solution is thus $M + M - 10\epsilon = 2M - 10\epsilon$. As ϵ approaches zero, the ratio of the optimal value to the value of NSS approaches $\frac{1}{2}$.

Running Time of NSS: Consider Algorithm 1. The first loop is executed B times. The number of sub-requests for each RRH is upper-bounded by N. Therefore, sorting sub-requests for all RRHs takes $O(BN \log(N))$ time. The second loop is executed B times. As the number of sub-requests in $\mathcal{U}_i$ of any

rrh_i is upper-bounded by N, the inner loop is executed at most N times. Finding vsr_j in the $\mathcal{U}_i$ of any rrh_i and removing it while keeping $\mathcal{U}_i$ sorted can be done in $O(\log(N))$ if an appropriate data structure such as a binary search tree is used to keep the sub-requests for any rrh_i. Overall, the running time of the second loop is $O(BN\log(N))$. Combining the running time of the first and the second loop yields the total running time of the algorithm as $O(BN\log(N))$.

D. Two Additional Heuristic Algorithms

Consider the example described to show the tightness of the bound for NSS in the previous sub-section. It is clear that the order of RRHs is what prevents NSS from obtaining a better performance. Based on this observation, we propose two other algorithms which consider RRHs in some specific order with the aim of avoiding such cases.

Description of BSS1:
BSS1 is a load balancing algorithm. It tries to distribute sub-requests among adjacent RRHs. To do so, all sub-requests are sorted in a non-increasing order at the beginning of the algorithm. Sub-requests are chosen for assignment according to this order. Consider the following definitions:

- *Current Load (cl_i):* For rrh_i, define the current load cl_i as the sum of the sizes of all sub-requests in $\mathcal{U}_i$ that have not been accommodated by rrh_i yet.
- *Remaining Capacity (rc_i):* For rrh_i, the remaining capacity rc_i indicates the available capacity of rrh_i which decreases by w_j each time a sub-request vsr_j is assigned to it.

When choosing vsr_j, the RRHs in $\mathcal{B}_j$ are compared with each other. Among them, rrh_i with $w_j \leq rc_i$ and the least ratio of $\frac{cl_i}{rc_i}$ is selected, and vsr_j is assigned to it. That is,

$$rrh_i = \underset{rrh_k : w_j \leq rc_k}{\arg\min} \frac{cl_k}{rc_k}. \tag{10}$$

Once some rrh_i is selected, cl_i and rc_i are updated accordingly. Then, the current loads of all other RRHs in $\mathcal{B}_j$ are updated as well. The operation of BSS1 is described in Algorithm 2.

Algorithm 2 Balancing Spectrum Sharing 1 (BSS1)

1: **Key step:** Sort sub-requests in a non-increasing order in terms of their requested weights (w_js)
2: **for** each vsr_j in sorted order **do**
3: $\quad rrh_i \leftarrow \underset{rrh_k : w_j \leq rc_k}{\arg\min} \frac{cl_k}{rc_k}$
4: $\quad$ Assign vsr_j to rrh_i
5: $\quad rc_i \leftarrow rc_i - w_j$
6: $\quad$ **for** each $rrh_k \in \mathcal{B}_j$ **do**
7: $\quad\quad cl_k = cl_k - w_j$
8: $\quad$ **end for**
9: **end for**

Running Time of BSS1: Sorting the sub-requests is done in $O(N\log(N))$. The outer for loop is executed N times. As the number of RRHs in $\mathcal{B}$ of each sub-request is a constant number, the loop body is executed in constant time. Overall, the running time of the outer for loop is $O(N)$. Therefore, the total running time of the algorithm is $O(N\log(N))$.

Description of BSS2:
In BSS2, at the beginning of the algorithm, the RRHs are sorted based on their ratio of the current load over the remaining capacity ($\frac{cl_i}{rc_i}$). Each time an RRH is selected in this order, the corresponding single knapsack problem is solved with the remaining assignable sub-requests. The steps of the algorithm are presented in Algorithm 3.

Algorithm 3 Balancing Spectrum Sharing 2 (BSS2)

1: **Key step:** Sort RRHs in a non-decreasing order in terms of the ratio of their current load and remaining capacity
2: **for** each rrh_i in the sorted order **do**
3: $\quad$ Solve the single knapsack problem for rrh_i with items in $\mathcal{U}_i$
4: $\quad$ Update rc_i
5: $\quad$ Update remaining capacity and $\mathcal{U}$ of all RRHs in $\mathcal{B}$ of any assigned sub-request
6: **end for**

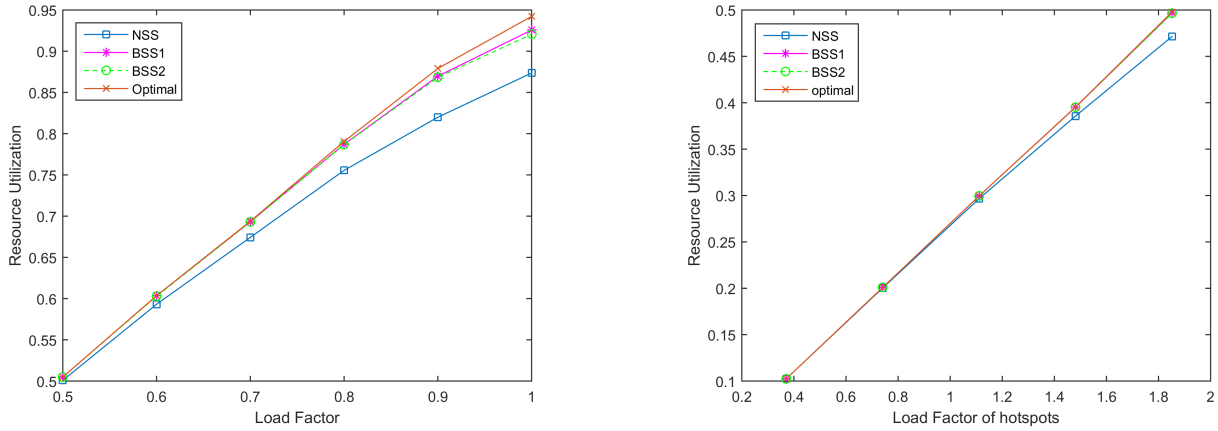
Running Time of BSS2: Sorting RRHs is done in $O(B\log(B))$. The for loop is executed B times. To solve a single knapsack problem for each RRH, sub-requests of each RRH are sorted in a non-increasing order in terms of their requested weights. Then, the sub-requests are chosen in this order until no more sub-requests can be accommodated by the RRH. The sorting process for each RRH, in step 3, takes $O(N\log(N))$ time, while the allocation process for each RRH is done in $O(N)$. Step 5 in the algorithm is executed in $O(N^2)$ time because the total number of assigned sub-requests is upper bounded by N. Also, the size of $\mathcal{U}$ of each RRH is upper bounded by N. The number of RRHs in $\mathcal{B}$ of any sub-request is constant. Other steps in the loop body have a constant running time. Overall, the running time of the algorithm is $O(BN^2 + B\log(B))$.

IV. Simulation Results

A. Simulation Setup

In this section, we present simulation results comparing the performances of our proposed algorithms and the optimal algorithm. In order to simulate the optimal algorithm, Gurobi solver [18] in CVX [19], which is a Matlab based optimization system is used. It is not possible to simulate the optimal algorithm for several time frames as it has exponential running time (recall that it is an NP-hard problem). Therefore, we simulate the optimal algorithm for only one time frame.

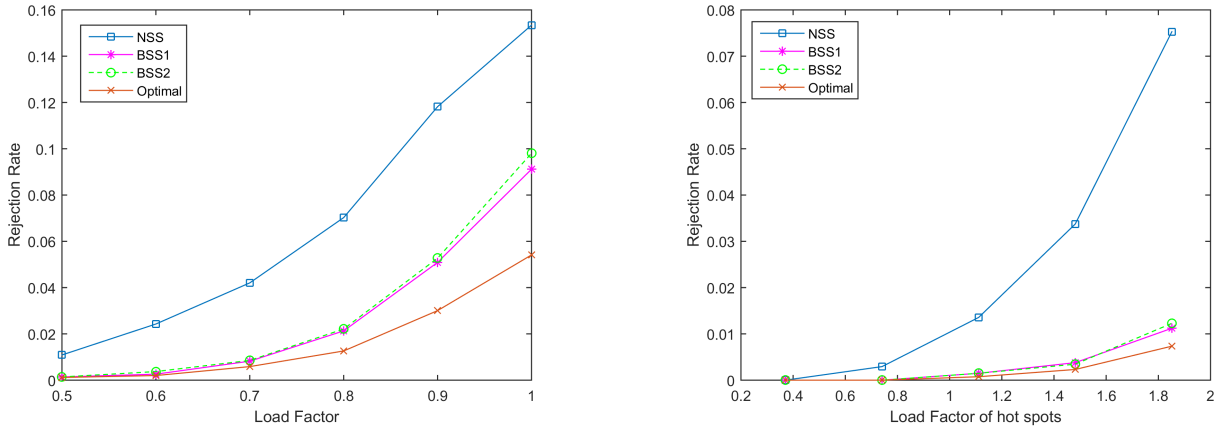
Our results are presented in two parts. In the first part, we compare the optimal algorithm and the proposed algorithms for one time frame. Once we establish the relative performances of our algorithms with respect to that of the optimal solution, in the second part, we focus on the performances of our algorithms over 500 time frames.



(a) Uniform scenario

(b) Non-uniform scenario

Fig. 5: Single time frame: Resource utilization of the optimal and proposed algorithms.



(a) Uniform scenario

(b) Non-uniform scenario

Fig. 6: Single time frame: Rejection rate of the optimal and proposed algorithms.

We use the utilization of the resource blocks and the rejection rate of sub-requests as the performance metrics. These performance metrics are defined as follows:

$$\textit{Utilization} = \frac{\text{number of allocated resource blocks}}{\text{number of available resource blocks}}.$$

$$\textit{Rejection rate} = \frac{\text{number of rejected sub-requests}}{\text{total number of sub-requests}}.$$

B. Network Configuration

The simulated cellular network has a regular layout. That is, RRHs are placed on a grid. There are 40 RRHs connected to the centralized processing pool in the Cloud-RAN. The arrival of sub-requests from virtual networks is simulated as a Poisson process. Note that our algorithms have no dependency on the arrival process or the distribution of resource requests in the network. These are used only to generate sample scenarios in simulations.

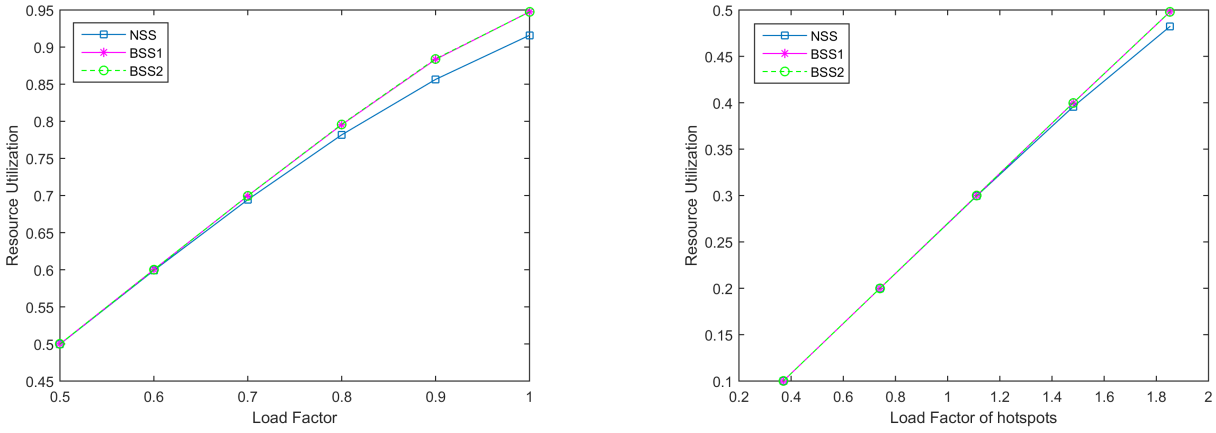
The average arrival rate changes from 40 to 400 sub-requests per time frame. We use the "load factor" to present our results, where the load factor for a time frame is defined as

$$\text{Load Factor} = \frac{\text{number of requested resource blocks}}{\text{number of available resource blocks}}.$$

The total number of resource blocks in each RRH is assumed to be 75. The size of each sub-request is uniformly distributed from 5 to 10 resource blocks. We consider two scenarios for the distribution of the sub-requests over the RRHs.

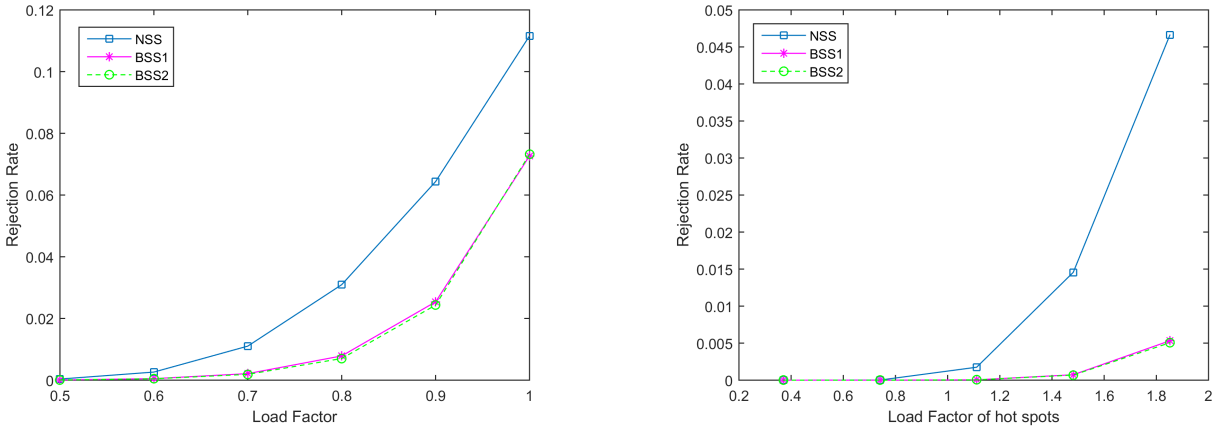
- *Uniform Scenario:* In this scenario, sub-requests are distributed uniformly over the RRHs.
- *Non-Uniform Scenario:* In this scenario, there are a few hot spots in the network. Hot spots are crowded areas such as down town area in working hours. To be concrete, 9 RRHs are selected as hot spots in the simulation.

In the non-uniform scenario, the hot spot RRHs have the highest load in the network. Therefore, the system performance is mostly affected by these areas, and we use the load



(a) Uniform scenario (b) Non-uniform scenario

Fig. 7: Multiple time frames: Resource utilization of the proposed algorithms.



(a) Uniform scenario (b) Non-uniform scenario

Fig. 8: Multiple time frames: Rejection rate of the proposed algorithms.

factor of the hot spots for the X-axis. Each point in the figures is the average of 20 simulation runs with different seeds.

Part I: All Algorithms over a Single Time Frame

As mentioned earlier, in this part, the optimal algorithm is simulated and compared with the three proposed algorithms for one time frame.

The results are shown in Figs. 5(a), 5(b), 6(a), and 6(b). It is observed that for light system loads, all algorithms achieve similar performances as available resources are sufficient to accommodate most sub-requests. The difference in performances emerges when the load factor is high. Even in this case, the performances of BSS1 and BSS2 are close to the performance of the optimal algorithm.

However, there is more noticeable difference between the performance of NSS and that of the optimal algorithm. The reason is that NSS does not distribute the load among adjacent RRHs evenly, which results in heavy load for some RRHs. Consequently, some RRHs suffer from high rejection rate, while some suffer from low resource utilization.

Another observation is that the resource utilization is lower in non-uniform scenario compared to the uniform scenario. This is true even for the optimal algorithm. The reason is that the load of some RRHs is very low in non-uniform scenario.

Part II: Heuristic Algorithms over Multiple Time Frames

In this part, we run NSS, BSS1 and BSS2 over 500 time frames. The objective is to study their performances over a long period of time. All simulation parameters are the same as in Part I.

Figs. 7(a) and 7(b) compare the resource utilization achieved by the three algorithms. As observed, BSS1 and BSS2 have better utilization than NSS. The difference between them is not significant because of the continuous arrival of sub-requests. In fact, different from Part I, in this set of simulations that are over several time frames, any sub-request vsr_j that enters the system waits in the queue for some time t_j. As a result, after

a few time frames from the start of the simulation, often there is enough load to utilize the available resources. Therefore, to compare them more accurately, one should compare the rejection rates of the algorithms as well. Figs. 8(a) and 8(b) compare the rejection rate of the three algorithms. As observed, the rejection rates of BSS1 and BSS2 are significantly lower than rejection rate of NSS because of the load balancing property of these two algorithms.

V. Conclusion

In this paper, we studied the problem of spectrum sharing in virtual radio access networks in a Cloud-RAN type network. The problem was formulated as an instance of the Multiple Knapsack with Assignment Restriction problem. We then developed a $\frac{1}{2}$-approximation algorithm for the problem called NSS. Two additional heuristic algorithms called BSS1 and BSS2 were also proposed to improve the worst-case performance of NSS by enabling load balancing among adjacent base stations. We simulated our proposed algorithms as well as the optimal algorithm under different network configurations. The simulation results confirmed that the performances of the proposed algorithms are close to that of the optimal algorithm. Moreover, BSS1 and BSS2 achieved a lower rejection rate than NSS because of their load balancing property.

An interesting extension of this work is to consider the problem of joint virtual network embedding and interference management.

References

[1] "Cisco visual networking index: Global mobile data traffic forecast update, 2014-2019," Cisco, Tech. Rep., 2015. [Online]. Available: www.cisco.com

[2] "Report of the spectrum efciency working group," Federal Communications Commission Spectrum Policy Task Force, Tech. Rep., 2002.

[3] C. Liang and F. R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 1, pp. 358–380, 2015.

[4] K. M. Park and C. K. Kim, "A framework for virtual network embedding in wireless networks," in *Proc. 4th International Conference on Future Internet Technologies*, 2009.

[5] J. van de Belt, H. Ahmadi, and L. E. Doyle, "A dynamic embedding algorithm for wireless network virtualization," in *Proc. IEEE VTC*, 2014, pp. 1–6.

[6] M. Yang, Y. Li, L. Zeng, D. Jin, and L. Su, "Karnaugh-map like online embedding algorithm of wireless virtualization," in *Proc. 15th International Symposium on Wireless Personal Multimedia Communications*, 2012, pp. 594–598.

[7] R. Kokku, R. Mahindra, Z. Honghai, and S. Rangarajan, "Cellslice: Cellular wireless resource slicing for active ran sharing," in *Proc. COMSNETS*, 2013, pp. 1–10.

[8] M. Yang, Y. Li, J. Liu, D. Jin, J. Yuan, and L. Zeng, "Opportunistic spectrum sharing for wireless virtualization," in *Proc. IEEE WCNC*, 2014, pp. 1803–1808.

[9] Y. Zaki, L. Zhao, C. Goerg, and A. Timm-Giel, "Lte mobile network virtualization," *Mobile Networks and Applications*, vol. 16, no. 4, pp. 424–432, 2011.

[10] H. Z. R. Kokku, R. Mahindra and S. Rangarajan, "Nvs: A substrate for virtualizing wireless resources in cellular networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1333–1346, 2012.

[11] "Network sharing: Architecture and functional description, v. 13.2.0," 3GPP TS 23.251, Tech. Rep., 2016.

[12] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud ran for mobile networks: A technology overview," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 1, pp. 405–426, 2015.

[13] A. Fischer, J. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[14] R. Mahindra, M. A. Khojastepour, H. Zhang, and S. Rangarajan, "Radio access network sharing in cellular networks," in *Proc. IEEE ICNP*, 2013, pp. 1–10.

[15] D. Lopez-Perez, I. Guvenc, G. D. L. Roche, M. Kountouris, T. Q. S. Quek, and J. Zhang, "Enhanced intercell interference coordination challenges in heterogeneous networks," *IEEE Wireless Communication*, vol. 18, no. 3, pp. 22–30, 2011.

[16] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004.

[17] M. Dawande, J. Kalagnanam, P. Keskinocak, R. Ravi, and F. Salman, "Approximation algorithms for the multiple knapsack problem with assignment restrictions," *Journal of Combinatorial Optimization*, vol. 4, no. 2, pp. 171–186, 2000.

[18] [Online]. Available: www.gurobi.com/

[19] [Online]. Available: http://cvxr.com/cvx/