# A Suggest-And-Improve RAN Slicing Strategy for Mobile Networks

Chengdong Yang*†, Tianyu Wang*, Zhe Xiao†, and Shaowei Wang*
*School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China
†Science and Technology on Communication Networks Laboratory, Shijiazhuang 050081, China
Email: cdyang@smail.nju.edu.cn, tianyu.alex.wang@nju.edu.cn, xiaozhe5401@126.com, wangsw@nju.edu.cn

***Abstract*—Network slicing is the key enabler to support vertical applications with diverse service level agreements and is expected to be inherited in the future mobile networks. However, due to the tricky inter-slice interference, the slicing strategy within the radio access network (RAN) domain is still considered to be a challenging problem, which needs to be optimized globally by taking the radio resource allocations of all cells and all slices into account. In this paper, we propose a low-complexity RAN slicing strategy by using the suggest-and-improve framework, where two high-level steps, i.e., the suggest step and the improve step, are performed alternatively. In the suggest step, a randomized method is utilized to find a candidate tuple consisting of a slice and a cell. In the improve step, a swapping operation is performed between the resource blocks of the specific slice and other slices within the specific cell. Simulation results show that the proposed suggest-and-improve algorithm outperforms the existing algorithms in terms of the total inter-slice interference as well as the average user throughput, while maintaining the computation time in the order of few milliseconds.**

***Index Terms*—Inter-slice interference, network slicing, radio access network, radio resource allocation, video streaming.**

## I. Introduction

With the rapid integration of vertical industries with mobile communication systems, the network-as-a-service paradigm has shown great potential in the follow-up evolution of 5G [1]. As compared to the existing industry-specific networks, where dedicated infrastructure and spectrum are deployed to build a physically private network, network slicing is envisioned as a better implementation that isolates multiple virtual end-to-end networks, referred to as slices, above a unified infrastructure to meet the various service requirements of different verticals. These slices include the eMBB slice supporting high resolution video streaming, the URLLC slice for life-critical applications, the MIoT slice for smart city applications, the V2X slice for vehicular communication, and the HMTC slice for industrial automation applications.

End-to-end network slicing can be divided into three relatively independent components corresponding to different network domains, i.e., the radio access network (RAN) slicing, the transport network slicing and the core network slicing. Among all three components, RAN slicing is considered to be the most challenging one due to the dynamic wireless environment and limited radio resources. In the early practice such as smart port and smart building [2], RAN slicing is implemented by using specific 5G quality-of-service (QoS) flows or dedicated bandwidth. The advantage is that they are easy to deploy in practice. However, these preliminary solutions cannot provide an efficient tradeoff between slice isolation and spectral efficiency. In the QoS flow solution, the slice performance decreases as the cell load increases, which violates the isolation requirement. In the dedicated bandwidth solution, the slice isolation requirement is fulfilled, while the spectral efficiency can be low for dynamic slice traffic.

In the literature, many efforts have been devoted to radio resource management for RAN slicing, whose main purpose is to achieve an efficient tradeoff between slice isolation and spectral efficiency [3]–[7]. However, due to the existence of inter-slice inter-cell interference, the existing methods proposed for the single-cell scenario cannot be extended to the more general case with multiple slices and multiple cells [8], [9]. On the one hand, inter-slice coordination is highly limited due to the isolation requirements and customized configurations of different slices, which implies inter-slice interference may cause unexpected impact on slice performance and severe data rate degradation of cell-edge users. On the other hand, it requires global resource block (RB) allocation to decrease inter-slice interference, which can be highly difficult considering the scale of practical networks. Thus, low-complexity inter-slice interference minimization solutions are of great importance for RAN slicing in multi-slice multi-cell networks.

In [10], the authors formulate RAN slicing as a quadratic programming problem with an indefinite matrix. The number of "linked" RBs is maximized to increase intra-slice coordinated transmissions and to decrease the inter-slice interference at the same time. The problem is proved to be NP-hard in general and the three approximation algorithms are proposed. In [11], a priority-based heuristic method is proposed by using the RB requests of different slices in difference cells, which intends to maximize the number of non-interference RBs. Numerical results show that it highly increases the orthogonality of RAN slices and achieves low computational complexity at the same time. In [12], the authors reuse the model in [10] and propose an iterative heuristic method by searching better RB allocations in local areas. Numerical results show that it increases the overall network throughput and achieves low computational complexity at the same time.

This work was supported in part by the National Natural Science Foundation of China under Grants 61931023 and U1936202.

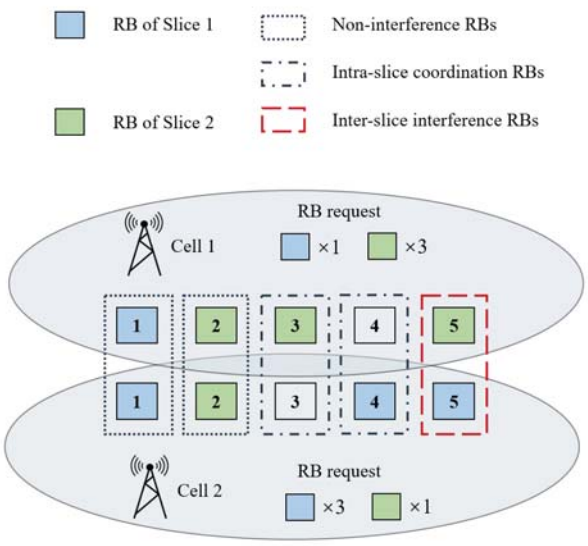


Fig. 1. Illustration of RAN slicing with 5 RBs, 2 slices and 2 cells.

In this paper, we consider the RAN slicing problem in multi-cell multi-slice networks. Specifically, we note that each allocated RB may interfere multiple neighboring cells, and thus, simply minimizing the number of RB pairs with inter-slice interference can be misleading. Here, we formulate a non-convex integer programming problem by minimizing the number of actually interfered RBs, and propose a heuristic algorithm based on the suggest-and-improve framework. The proposed method utilizes the inter-slice interference level as a heuristic to suggest a specific slice in a specific cell, and then adjusts the local RB allocation to improve the overall inter-slice interference performance. Numerical results show that the proposed algorithm outperforms the state-of-the-art algorithms in terms of both the total inter-slice interference and the average user throughput, while at the same time, maintaining low computational time of few milliseconds.

## II. System Model

We consider the downlink of a RAN with $S$ slices and $N$ cells, the sets of which are denoted as $\mathcal{S}$ and $\mathcal{N}$, respectively. We note that each slice may cover multiple cells and each cell may involve multiple slices. We denote by $\boldsymbol{A} \in \{0,1\}^{N\times N}$ as the adjacent matrix of cells, in which $a_{i,j} = 1$ if $i \neq j$ and the coverage areas of cell $i$ and cell $j$ overlap each other, and $a_{i,j} = 0$ otherwise. For any cell $n \in \mathcal{N}$, we denote by $\mathcal{B}_n = \{i \in \mathcal{N} \mid a_{n,i} = 1\}$ as the set of its neighboring cells and by $B_n = |\mathcal{B}_n|$ the number of its neighboring cells. The spectrum is equally divided into $M$ RBs, each of which represents the minimum resource unit that can be requested by any slice in any cell. The set of all RBs is denoted as $\mathcal{M}$. In Fig. 1, we show an illustrative example with $M = 5$ RBs, $S = 2$ slices and $N = 2$ cells.

We denote by $L_{s,n}$ as the number of requested RBs of slice $s$ in cell $n$, which is a long-term estimation of the total traffic demand of the corresponding users. We denote by $L_n^{\text{cell}}$ as the total number of requested RBs of all slices in cell $n$, i.e., $L_n^{\text{cell}} = \sum_{s=1}^{S} L_{s,n}$. We denote by $L_s^{\text{slice}}$ as the total number of requested RBs of slice $s$ in all cells, i.e., $L_s^{\text{slice}} = \sum_{n=1}^{N} L_{s,n}$. Obviously, we have $L_n^{\text{cell}} \leq M, 1 \leq n \leq N$. We denote by $L = \sum_{n=1}^{N} L_n^{\text{cell}}/M$ as the traffic load of the considered RAN. In the illustrative example, cell 1 requests $L_{1,1} = 1$ RBs for slice 1 and $L_{1,2} = 3$ RBs for slice 2, while cell 2 requests $L_{2,1} = 3$ RBs for slice 1 and $L_{2,2} = 1$ RBs for slice 2. The traffic load is given by $L = 0.8$.

We denote by $\boldsymbol{p}_{s,n} \in \{0,1\}^M$ as the RB allocation of slice $s$ in cell $n$, in which $\boldsymbol{p}_{s,n}(m) = 1$ if the $m$-th RB of cell $n$ is allocated to slice $s$, and $\boldsymbol{p}_{s,n}(m) = 0$ otherwise. We denote by $\boldsymbol{q}_s = [\boldsymbol{p}_{s,1}^T, \boldsymbol{p}_{s,2}^T, \ldots, \boldsymbol{p}_{s,N}^T]^T$ as the RB allocation of slice $s$, and by $\boldsymbol{x} = [\boldsymbol{q}_1^T, \boldsymbol{q}_2^T, \ldots, \boldsymbol{q}_S^T]^T$ as the overall RB allocation. Thus, $\boldsymbol{x}$ is an $SNM$-dimensional column vector with all its elements belonging to $\{0,1\}$, i.e.,

$$\boldsymbol{x} \in \{0,1\}^{SNM}. \tag{1}$$

In the illustrative example, the overall RB allocation is given by $\boldsymbol{x} = (1,0,0,0,0,1,0,0,1,1,0,1,0,1,1,0,1,0,0,0)^T$ and it fulfills all RB requests.

For a feasible RB allocation $\boldsymbol{x}$, any RB $m$ of any cell $n$ can only be allocated to at most one slice. Thus, we have

$$\boldsymbol{x}^T \boldsymbol{y}_{n,m} \leq 1, \tag{2}$$

where $\boldsymbol{y}_{n,m} \in \{0,1\}^{SNM}$ is the vector for the summation of all slices for RB $m$ in cell $n$, i.e., $\boldsymbol{y}_{n,m}(k) = 1$ if $k \equiv (n-1)M + m \pmod{NM}$, and $\boldsymbol{y}_{n,m}(k) = 0$ otherwise.

Also, for a feasible RB allocation $\boldsymbol{x}$, the number of RBs requested by any slice $s$ in any cell $n$ must be fulfilled. Thus, we have

$$\boldsymbol{x}^T \boldsymbol{z}_{s,n} = L_{s,n}, \tag{3}$$

where $\boldsymbol{z}_{s,n} \in \{0,1\}^{SNM}$ is the vector for the summation of all RBs for slice $s$ in cell $n$, i.e., $\boldsymbol{z}_{s,n}(k) = 1$ if $k \geq (s-1)NM + (n-1)M + 1$ and $k \leq (s-1)NM + nM$, and $\boldsymbol{z}_{s,n}(k) = 0$ otherwise.

According to the specific RB allocation $\boldsymbol{x}$, the RBs can be classified into three RB types, i.e., the non-interference RB, the intra-slice coordination RB and the inter-slice interference RB. For the non-interference RB, the corresponding RB is not occupied by any slice in any of its neighboring cells. For the intra-slice coordination RB, the corresponding RB is only occupied by the same slice in its neighboring cells, for which coordinated multi-point techniques can be applied to avoid intra-slice interference. For the inter-slice interference RB, the corresponding RB is occupied by some other slice in one of its neighboring cells, in which case the inter-slice interference may occur. In the illustrative example, RBs 1 and 2 are non-interference RBs, RBs 3 and 4 are intra-slice coordination RBs, and RB 5 is an inter-slice interference RB.

Here, we focus on minimizing the total number of inter-slice interference RBs. For any RB $m$ in cell $n$ allocated to slice $s$, we have $x_{s,n,m} = 1$. For any neighboring cell $n'$, we have $a_{n,n'} = 1$. Thus, the corresponding RB suffers from inter-slice

interference from cell $n'$ if and only if $\sum_{s' \neq s} x_{s',n',m} = 1$. Let us consider the matrix

$$\boldsymbol{Q} = \boldsymbol{I}_S \otimes (\boldsymbol{A} \otimes \boldsymbol{I}_M), \tag{4}$$

where $\otimes$ stands for the Kronecker product and $\boldsymbol{I}_k$ is the $k \times k$ identity matrix. Therefore, given any RB allocation $\boldsymbol{x}$, the total amount of inter-slice interference can be written as $\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x}$. We note that the inter-slice interference RB are counted multiple times when there are multiple cells causing inter-slice interference. Thus, the total number of inter-slice interference RBs is given by

$$f(\boldsymbol{x}) = \min\left\{\boldsymbol{x}^T \boldsymbol{Q}, \mathbf{1}\right\} \boldsymbol{x}, \tag{5}$$

where $\min\{\cdot,\cdot\}$ returns the minimum value in each dimension and $\mathbf{1}$ is an $SNM$-dimensional row vector with all its elements being equal to 1.

Therefore, the overall inter-slice interference minimization problem is formulated as

$$\min_{\boldsymbol{x}} \ f(\boldsymbol{x}), \tag{6a}$$

$$\text{s.t. } \boldsymbol{x} \in \{0,1\}^{SNM}, \tag{6b}$$

$$\boldsymbol{x}^T \boldsymbol{y}_{n,m} \leq 1, 1 \leq n \leq N, 1 \leq m \leq M, \tag{6c}$$

$$\boldsymbol{x}^T \boldsymbol{z}_{s,n} = L_{s,n}, 1 \leq s \leq S, 1 \leq n \leq N. \tag{6d}$$

We note that the objective function (6a) is non-convex and constraint (6b) is an integer constraint. Thus, problem (6) is a non-convex integer programming problem, which is NP-hard in general. In the following section, we propose a heuristic RAN slicing algorithm by using the suggest-and-improve framework, which utilizes problem-dependent heuristics to search for high performance solutions.

## III. Suggest-And-Improve Algorithm

The proposed suggest-and-improve algorithm consists of three major components, i.e., the initialization method to find a feasible RB allocation as the start point, the suggest method to select a candidate slice and a candidate cell, and the improve method to adjust the RB allocation of the specific slice in the specific cell. The suggest and improve methods are performed alternatively to decrease the overall inter-slice interference. The implementation details are given as follows.

### A. Initialization

The initial RB allocation $\boldsymbol{x}_0$ is constructed by sequentially fulfilling the RB requests $\{L_{s,n}\}_{s,n}$ of all slices in all cells. Specifically, we set $\boldsymbol{x}_0 = \mathbf{0}$ and consider the RB requests with an outer loop of slices and an inner loop of cells. The slices are sorted in a decreasing order of the number of requested RBs, i.e., $s_1, s_2, \ldots, s_S \in \mathcal{S}$ where $L_{s_i}^{\text{slice}} \geq L_{s_j}^{\text{slice}}$ for any $1 \leq i < j \leq S$. The cells are sorted in a decreasing order of the number of neighboring cells, i.e., $n_1, n_2, \ldots, n_N \in \mathcal{N}$ where $B_{n_i} \geq B_{n_j}$ for any $1 \leq i < j \leq N$.

For any index pair $(i,j)$, we consider the RB request $L_{s_i,n_j}$ of slice $s_i$ in cell $n_j$ with the instant RB allocation $\boldsymbol{x}_0$. The set of unallocated RBs in cell $n_j$ is given by

$$\mathcal{M}_j(\boldsymbol{x}_0) = \left\{ m \in \mathcal{M} \;\middle|\; \boldsymbol{x}_0^T \boldsymbol{y}_{n_j,m} = 0 \right\}. \tag{7}$$

For any $m \in \mathcal{M}_j(\boldsymbol{x}_0)$, the total number of inter-slice interference caused by the allocation of RB $m$ in cell $n_j$ to slice $s_i$ is given by

$$C_{i,j}(m; \boldsymbol{x}_0) = \sum_{n \in \mathcal{B}_{n_j}} \sum_{s \neq s_i} \boldsymbol{x}_0^T \boldsymbol{e}_{s,n,m}, \tag{8}$$

where $\boldsymbol{e}_{s,n,m}$ is an $SNM$-dimensional unit vector indicating whether RB $m$ in cell $n$ is allocated to slice $s$, i.e., $\boldsymbol{e}_{s,n,m}(k) = 1$ if $k = (s-1)NM + (n-1)M + m$, and $\boldsymbol{e}_{s,n,m}(k) = 1$ otherwise. Thus, $C_{i,j}(m; \boldsymbol{x}_0)$ can be seen as the cost of allocating RB $m$ in cell $n_j$ to slice $s_i$. We select the RB with the minimum cost, i.e.,

$$m^* = \underset{m \in \mathcal{M}_j(\boldsymbol{x}_0)}{\arg\min} \ C_{i,j}(m; \boldsymbol{x}_0). \tag{9}$$

Then, RB $m^*$ in cell $n_j$ is allocated to slice $s_i$. Thus, we have

$$\boldsymbol{x}_0 \left[(s_i - 1)NM + (n_j - 1)M + m^*\right] = 1. \tag{10}$$

For any $L_{s_i,n_j} > 0$, the allocation is performed $L_{s_i,n_j}$ times until all requested RBs of slice $s_i$ in cell $n_j$ are fulfilled. We note that constraints (6b) and (6c) are strictly satisfied during the construction of $\boldsymbol{x}_0$, and after $SN$ iterations when all RB requests $\{L_{s_i,n_j}\}_{i,j}$ are fulfilled, constraint (6d) is satisfied. Thus, the outcome $\boldsymbol{x}_0$ is a feasible solution of problem (6).

### B. Suggest Method

Given the current RB allocation $\boldsymbol{x}$, the suggest method returns a tuple $(s,n)$ consisting of the suggested slice $s \in \mathcal{S}$ and the suggested cell $n \in \mathcal{N}$. Specifically, for any slice $s \in \mathcal{S}$ in any cell $n \in \mathcal{N}$, the set of allocated RBs is given by

$$\mathcal{M}_{s,n}(\boldsymbol{x}) = \left\{ m \in \mathcal{M} \;\middle|\; \boldsymbol{x}^T \boldsymbol{e}_{s,n,m} = 1 \right\}, \tag{11}$$

and the total number of experienced inter-slice interference is given by

$$R_{s,n}(\boldsymbol{x}) = \sum_{m \in \mathcal{M}_{s,n}(\boldsymbol{x})} \sum_{n \in \mathcal{B}_{n_j}} \sum_{s \neq s_i} \boldsymbol{x}^T \boldsymbol{e}_{s,n,m}. \tag{12}$$

According to the weights given by $R_{s,n}(\boldsymbol{x})$, we define a discrete conditional probability distribution $p(s,n|\boldsymbol{x})$ of tuple $(s,n)$ given $\boldsymbol{x}$, i.e.,

$$p(s,n|\boldsymbol{x}) = \frac{R_{s,n}(\boldsymbol{x})}{\sum_{s'=1}^{S} \sum_{n'=1}^{N} R_{s',n'}(\boldsymbol{x})}. \tag{13}$$

In each iteration, the suggest method samples a tuple $(s^*, n^*)$ from the distribution $p(s,n|\boldsymbol{x})$, which implies a specific slice in a specific cell with high inter-slice interference. The tuple is then forwarded to the following improve method to reduce the inter-slice interference by adjusting its RB allocation.

### C. Improve Method

Given the current RB allocation $\boldsymbol{x}$ and the suggested tuple $(s^*, n^*)$, the improve method intends to decrease the total number of inter-slice interference RBs by swapping the RB allocated to the suggested tuple with other RBs. Specifically, we randomly select an RB $m$ from the set $\mathcal{M}_{s^*,n^*}(\boldsymbol{x})$ of RBs allocated to slice $s^*$ in cell $n^*$, and another RB $m'$ from the

**Algorithm 1:** Suggest-And-Improve for RAN Slicing

**Input**: $S, N, M, \boldsymbol{A}, \{L_{s,n}\}_{s,n}$
**Output**: RB allocation $\boldsymbol{x}$

```
/*Initialization*/
x_0 ← 0;
for i ∈ [1, S] do
    for j ∈ [1, N] do
        for k ∈ [1, L_{s_i,n_j}] do
            Calculate RB m* as in (9);
            Update RB allocation x_0 as in (10);
        end
    end
end
x ← x_0;
for t ∈ [1, T] do
    /*Suggest*/
    Calculate probabilities p(s, n|x) as in (13);
    Sample (s*, n*) from distribution p(s, n|x);
    /*Improve*/
    Calculate new RB allocation x' as in (14) and (15);
    if f(x') < f(x) then
        x ← x';
    end
end
```

set $\mathcal{M} - \mathcal{M}_{s^*,n^*}(\boldsymbol{x})$ of the rest RBs. We swap the allocation of RBs $m$ and $m'$, and obtain a new RB allocation $\boldsymbol{x}'$. If RB $m'$ is originally unoccupied, we have

$$\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{e}_{s^*,n^*,m} + \boldsymbol{e}_{s^*,n^*,m'}. \tag{14}$$

If RB $m'$ is originally occupied by another slice $s' \neq s$, i.e., $\boldsymbol{x}[(s'-1)NM + (n^*-1)M + m'] = 1$, we have

$$\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{e}_{s^*,n^*,m} - \boldsymbol{e}_{s',n^*,m'} + \boldsymbol{e}_{s^*,n^*,m'} + \boldsymbol{e}_{s',n^*,m}. \tag{15}$$

If the number of inter-slice interference RBs is decreased after the swapping operation, i.e., $f(\boldsymbol{x}') < f(\boldsymbol{x})$, we update the RB allocation as $\boldsymbol{x} = \boldsymbol{x}'$. Otherwise, the RB allocation $\boldsymbol{x}$ stays unchanged. Note that the swapping operation ensures $\boldsymbol{x}'$ is also a feasible solution of problem (6). Thus, the RB allocation is still feasible after multiple iterations of alternative suggest and improve steps.

### D. Convergence

As the number of inter-slice interference RBs decreases monotonically as the iterations proceed, the proposed algorithm converges when there exists no suggested tuple that can be improved by any swapping operation. Since the number of inter-slice interference RBs is bounded between 0 and $NM$, the proposed algorithms converges in no more than $NM$ swapping operations. However, due to the randomness of the proposed algorithm, a large number of iterations may be required to reach the convergence point. In practice, we can set a large iteration number $T$ to achieve a tradeoff between the network performance and computation complexity. We summarize the proposed suggest-and-improve RAN slicing algorithm in **Algorithm** 1.

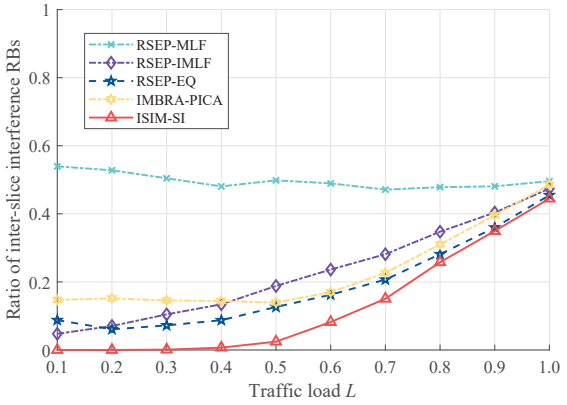


Fig. 2. Ratio of inter-slice interference RBs as a function of traffic load $L$.

## IV. Numerical Results

We consider a network with $N = 5$ cells and $S = 3$ slices. The base stations are dropped randomly in a $500 \times 500$ m square area with a uniform cell radius of 250 m. The total 100 MHz bandwidth is equally divided into $M = 20$ RBs. We assume the total number of requested RBs is equal for all cells, i.e., $L_1^{\text{cell}} = L_2^{\text{cell}}, \ldots, L_N^{\text{cell}}$, and the corresponding RB requests are distributed randomly among all slices. We assume there are 5 users of each slice in each cell, which are distributed randomly and follow the full buffer model. The proportional fair scheduler is applied for users of a specific slice within a specific cell, and coordinated transmissions are applied for cell-edge users within the same slice. We consider a typical urban propagation environment, where the pathloss exponent is 3.76 and the standard deviation of log-normal shadowing is 3 dB [13]. The small scale fading is assumed to be Rayleigh. The transmit power is uniformly given by 20 dBm,

We compare the proposed suggest-and-improve algorithm for inter-slice interference minimization (ISIM-SI) with four state-of-the-art algorithms, i.e., the RSEP-EQ and RSEP-MLF algorithms proposed in [10], the RSEP-IMLF algorithm proposed in [12] and the IMNRA-PICA algorithm proposed in [11]. We note that the RSEP-EQ algorithm is based on convex quadratic programming, and the other three algorithms are heuristic algorithms. The maximum iteration number of the proposed algorithm is given by $T = 5SN$.

In Fig. 2, we show the ratio of inter-slice interference RBs as a function of traffic load $L$. Intuitively, as the traffic load increases, the number of total requested RBs increases and the competition among slices are intensified, which makes it more difficult to avoid inter-slice interference. Thus, the ratio of inter-slice interference RBs increases with the traffic load. Specifically, the curves increase slowly in the low traffic load region as the radio resources are relatively sufficient. While, as the traffic load surpasses a certain threshold, the RB allocations of neighboring cells cannot be well coordinated

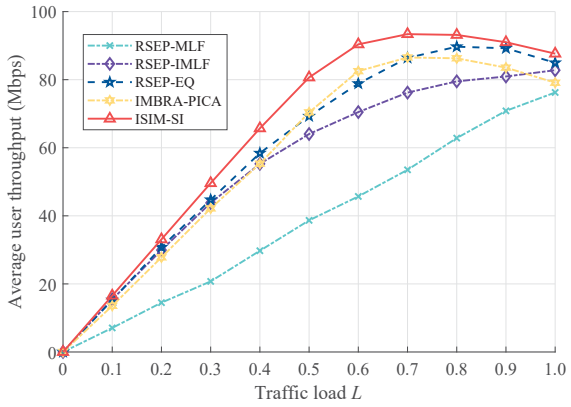


Fig. 3. Average user throughput as a function of traffic load $L$.

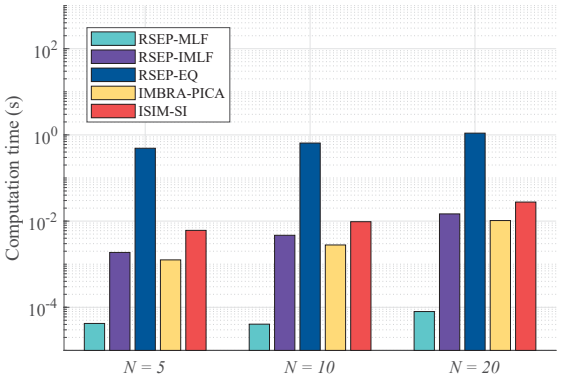


Fig. 4. Computation time with different number $N$ of cells.

and the curves increase rapidly. As we can see, the proposed ISIM-SI algorithm outperforms the benchmark algorithms in the entire region. We note that the RSEP-MLF algorithm has a slightly decreasing curve as it mainly focus on improving intra-slice cooperation, which may not fully utilize the entire bandwidth in the low traffic load region.

In Fig. 3, we show the average user throughput as a function of traffic load $L$. The average user throughput is fundamentally determined by the amount of allocated RBs as well as the interference level. As intra-cell and intra-slice interference are strictly avoided by packet scheduling, the interference level is fully determined by the inter-slice interference. In the low traffic load region, the inter-slice interference keeps low as shown in Fig. 2, while the number of allocated RBs increases linearly as the traffic load increases. Thus, the average user throughput increases rapidly. In the high traffic load region, the inter-slice interference increases rapidly as shown in Fig. 2, which offsets the gains from the increased number of allocated RBs. Thus, the growth of the average user throughput slows down and even declines at extreme high traffic loads. As we can see, the proposed ISIM-SI algorithm still outperforms the benchmark algorithms in the entire region.

In Fig. 4, we show the computation time of all considered algorithms on a Core i7 CPU with $N = 5, 10, 20$, respectively. As the number of cells increases, the computation time of both the mathematical programming and the heuristic algorithms increases. As we can see, the mathematical programming algorithm, i.e., the RSEP-EQ algorithm, requires computation time in the order of several seconds. While, the heuristic algorithms, including the RSEP-IMLF, IMNRA-PICA and the proposed ISIM-SI algorithms, only requires computation time in the order of several milliseconds. We note that even though the RSEP-MLF algorithm has sub-millisecond computation time, it suffers from a significant performance degradation as compared to the other algorithms.

## V. Conclusion

In this article, we considered the inter-slice interference minimization problem in multi-cell multi-slice networks. First, we analyze the objective and constraints of RAN slicing and formulate a non-convex integer programming problem. Then, we propose a suggest-and-improve RAN slicing algorithm to improve the global performance by iteratively performing local improvements. In addition, we prove the proposed algorithm converges in finite iterations. At last, simulation results verify the superior performance of the proposed algorithm in terms of the total inter-slice interference and the average user throughput as compared to other existing methods, as well as its low computation time in typical RAN settings.

## References

[1] H. Zhang *et al.*, "Network slicing based 5G and future mobile networks: Mobility, resource management, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 138–145, Aug. 2017.
[2] GSMA, "Powered by SA: Smart port MEC security application," White Paper, 2020.
[3] T. Guo and A. Suárez, "Enabling 5G RAN slicing with EDF slice scheduling," *IEEE Trans. Veh. Tech.*, vol. 68, no. 3, pp. 2865–2877, Mar. 2019.
[4] T. Wang and S. Wang, "Inter-slice radio resource allocation: an online convex optimization approach," *IEEE Wireless Commun.*, vol. 28, no. 5, pp. 171–177, Sept. 2021.
[5] V. Sciancalepore *et al.*, "RL-NSB: Reinforcement learning-based 5G network slice broker," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1543–1557, Aug. 2019.
[6] T. Wang and S. Wang, "Online convex optimization for efficient and robust inter-slice radio resource management," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 6050–6062, Jun. 2021.
[7] L. Shen *et al.*, "Proactive proportional fair: A novel scheduling algorithm based on future channel information in OFDMA systems," in *Proc. IEEE ICCC'19*, Changchun, China, Aug. 2019.
[8] S. D'Oro *et al.*, "Toward operator-to-waveform 5G radio access network slicing," *IEEE Commun. Mag.*, vol. 58, no. 4, pp. 18–23, Apr. 2020.
[9] J. Li *et al.*, "A hierarchical soft RAN slicing framework for differentiated service provisioning," *IEEE Wireless Commun.*, vol. 27, no. 6, pp. 90–97, Apr. 2020.
[10] S. D'Oro *et al.*, "The slice is served: Enforcing radio access network slicing in virtualized 5G systems," in *Proc. IEEE INFOCOM'19*, Paris, France, Apr. 2019.
[11] H. Li *et al.*, "An interference minimization-based RAN slicing strategy in 5G systems," in *Proc. IEEE ISWCS'21*, Berlin, Germany, Sept. 2021.
[12] S. D'Oro *et al.*, "Coordinated 5G network slicing: How constructive interference can boost network throughput," *IEEE/ACM Trans. Netw.*, vol. 29, no. 4, pp. 1881–1894, Apr. 2021.
[13] Björnson *et al.*, "Massive MIMO networks: Spectral, energy, and hardware efficiency," *Found. Trends Signal Process.*, vol. 11, no. 3-4, pp. 154–655, Nov. 2017.