Power Coverage Optimization in Cellular Radio Networks Using Neural Architecture Search

Juan Zhu*†, 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: juanzhu@smail.nju.edu.cn, tianyu.alex.wang@nju.edu.cn, xiaozhe5401@126.com, wangsw@nju.edu.cn

***Abstract*—Due to the complex propagation environment, it is difficult to find the optimal antenna configuration that ensures the best coverage of cellular networks. Most existing work adopts heuristic methods, where the antenna configuration is evaluated and updated by calculating the coverage of the entire area iteratively. However, in real networks with a large number of antennas and highly complex ray-tracing models, the heavy computational burden caused by repetitive coverage calculation is usually unacceptable. In this paper, we propose a deep learning approach that straightly infers the antenna configuration from the raw data given by the city map. Specifically, we adopt the emergent neural architecture search to automatically design a specific neural network for the considered power coverage optimization problem. Numerical results show that the discovered neural architecture greatly outperforms the conventional hand-crafted architectures.**

***Index Terms*—Deep learning, network planning, neural architecture search, power coverage optimization.**

I. Introduction

Coverage optimization is one of the fundamental problems in mobile network planning, which aims to find the configuration that maximizes the area with a minimum received power, referred to as power coverage optimization, or a minimum signal-to-interference-and-noise-ratio, referred to as capacity coverage optimization [1]. Depending on the specific requirements, the configuration parameters may include the number, position and transmit power of base stations [2], and the number, type, downtilt and azimuth of antennas [3]. The corresponding coverage performance highly impacts a variety of key performance indicators for mobile users, such as user data rate, call drop rate and handover failure.

Due to the complex propagation environment and the large number of configurable parameters, coverage optimization is considered to be highly challenging, and it becomes more challenging in future wireless networks where small cells with higher carrier frequencies are densely deployed. In fact, coverage optimization belongs to a class of NP-hard problems, which is computationally difficult even with medium sizes [1]. Thus, most existing work resorts to heuristic methods, including simulated annealing [4], tabu search [5], evolutionary algorithms [6], ant colony algorithms [7] and approximation algorithms [2]. Some other work approximates the original NP-hard problem into other mathematical forms with more sophisticated solutions. In [8], the coverage optimization problem is analyzed under the multi-arm bandit framework, where the proposed algorithm is proved to converge to the optimal arm with bounded regret. In [9], a numerical optimization framework is utilized, where the classic stochastic gradient descent algorithm is utilized to achieve local optimum with low computational complexity.

Despite the conciseness and intuitiveness, the above algorithms suffer from the computational burden caused by the repetitive calculation of complex ray-tracing propagation models in real networks. To reduce the computational effort, deep learning based methods are recently introduced to estimate the coverage performance directly from the environment information. In [10], a two-step method based on reinforcement learning is proposed to adjust the antenna parameters iteratively according to the estimated coverage performance given by a deep neural network. In [11], an auto-encoder structure is adopted to extract semantic features from satellite imagery to divide a geographical terrain into different segments, such that each segment can employ an environment-specific empirical model with fine-tuned propagation parameters.

We note that the neural networks employed by existing work are originally engineered for traditional machine learning tasks such as image recognition and natural language processing, and they may not have the most suitable architecture for the considered coverage optimization problem. Traditionally, the search for suitable architectures is in itself a time-consuming and error-prone task, though task-specific architecture modifications may result in significant gains [12]. However, there has been an insurgence in recent research efforts in automatically searching for problem-dependent neural architectures without any human invention, which is referred to as neural architecture search (NAS) [13]. Already by now, NAS methods have outperformed manually designed architectures on various machine learning tasks [14].

In this paper, we focus on the power coverage optimization problem of a single small cell, where the antenna azimuth and downtilt are adjusted to maximize the area with a minimum received power. Specifically, we utilize the state-of-the-art NAS technology to automatically design a suitable neural architecture of the considered problem, which takes the map information as its input and directly outputs the suggested

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

angles of antenna azimuth and downtilt. Numerical results show that the discovered neural architecture greatly differs from the hand-crafted architectures, and greatly outperforms them in terms of model accuracy and online complexity.

II. Power Coverage Optimization with Deep Neural Networks

In this section, we formulate the considered power coverage optimization as a machine learning task, where a deep neural network is utilized to infer the optimal azimuth and downtilt angles given the propagation environment. Specifically, we construct a problem-dependent dataset by using a commercial radio propagation software with real city maps and a radiation pattern from a commercial 5G antenna, and then train the deep neural network with a loss function measuring the difference between the output configuration and the optimal configuration. We note that the architecture of the deep neural network is not specified, as it will be automatically designed by NAS in the next section.

A. Power Coverage Ratio

Consider a single antenna sitting at the center of a square area. We denote by h the antenna height and then the antenna location is given by $\boldsymbol{r}_0 = (0, 0, h)$. We denote by θ and ϕ the antenna's azimuth and downtilt angles in the global coordinate system, respectively, and by α and β the antenna's azimuth and elevation angles in the local coordinate system, respectively. The corresponding directional gain is given by $G(\alpha, \beta)$ and the transmit power is given by P_t.

We divide the service area into K equal grids and assume the coverage performance of each grid is represented by the received signal power at the center point. For any grid point i located at $\boldsymbol{r}_i = (x_i, y_i, z_i)$, the transmission direction in the local coordinate system is then given by

$$\alpha_i = \theta - \arctan\frac{x_i}{y_i}, \tag{1}$$

and

$$\beta_i = \phi - \arctan\frac{h - h_i}{\sqrt{x_i^2 + y_i^2}}. \tag{2}$$

Thus, the received signal power of grid point i is given by

$$P_i = P_t \cdot G(\alpha_i, \beta_i) \cdot PL(\boldsymbol{r}_0, \boldsymbol{r}_i; \mathcal{E}), \tag{3}$$

where $PL(\boldsymbol{r}_0, \boldsymbol{r}_i; \mathcal{E})$ represents the pathloss from $\boldsymbol{r}_0$ to $\boldsymbol{r}_i$ with propagation environment $\mathcal{E}$, representing the geometrical information of terrain and buildings of the service area.

We define a threshold P_0 for the received signal power. A grid is covered if and only if the received signal power at the grid point is above the threshold. Thus, the power coverage ratio is given by

$$R(\theta, \phi) = \frac{1}{K}\sum_{i=1}^{K} \mathbf{1}(i)_{P_i \geq P_0}, \tag{4}$$

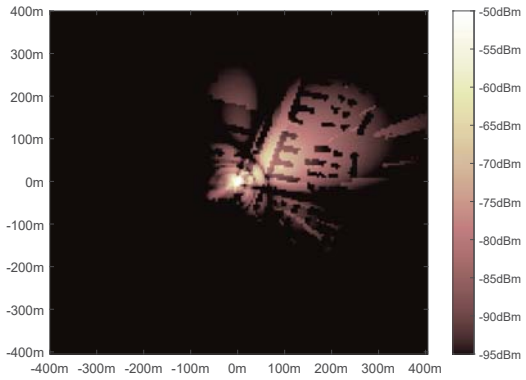


Fig. 1. Illustration of the power coverage of a square area with 161×161 grids, where $\theta = 60^\circ$ and $\phi = 15^\circ$.

where $\mathbf{1}(x)$ is an indicator function satisfying $\mathbf{1}(i) = 1$ if $P_i \geq P_0$ and $\mathbf{1}(i) = 0$ otherwise. Thus, the optimal antenna configuration can be written as

$$(\theta^*, \phi^*) = \arg\max_{\theta,\phi} R(\theta, \phi). \tag{5}$$

Due to the complex propagation environment and the complex propagation model, it is usually difficult to find the optimal antenna configuration. In the following subsections, we adopt a machine learning model to address the power coverage optimization problem, which takes $\mathcal{E}$ and h as the model input and directly outputs the antenna configuration.

B. Dataset Construction

We construct a dataset by using a commercial radio propagation software *WinProp* with a real city map and a 5G antenna radiation pattern [15]. To reduce the computational efforts, the number of feasible configurations is restricted by 60. Specifically, we have $\theta = n \times 30^\circ$ and $\phi = m \times 5^\circ$, where $0 \leq n \leq 11$ and $0 \leq m \leq 4$ are integers. The city map generates 12000 square areas of 805 m $\times$ 805 m, each of which is divided into 161×161 grids of 5 m $\times$ 5 m. In Fig. 1, we provide an illustrative example of the power coverage of a square area with $\theta = 60^\circ$ and $\phi = 15^\circ$.

For each square i, the map feature is given by a tensor $\boldsymbol{s}^{(i)} \in \mathbb{R}^{3*161*161}$, where the first 161×161 channel represents the terrain height of all grids, and the second 161×161 channel represents the building heights of all grids, and the third 161×161 channel is simply the product of an all-one matrix J_{161} and the antenna height h. The label $\boldsymbol{l}^{(i)} \in \mathbb{R}^{60}$ is given by a one-hot vector representing the index of the optimal antenna configuration given by exhaustive search. Therefore, the dataset can be written as

$$\mathcal{D} = \left\{ (\boldsymbol{s}^{(i)}, \boldsymbol{l}^{(i)}) \mid 1 \leq i \leq 12000 \right\}. \tag{6}$$

C. Model Training

The dataset is divided into a training set of 5000 samples, a validation set of 5000 samples and a test set of 2000 samples.

The training set is used to fit the model parameters according to a loss function representing the difference between the true label and the inferred label. The validation set is used to search for the suitable neural network architecture. The test set is used to evaluate the performance of the final model. We adopt the cross-entropy loss function to penalize the model when the output is different from the true label. We denote by $\hat{\boldsymbol{l}}^{(i)}$ as the model output of sample i. Thus, the total loss of a set of M samples is given by

$$f_M(\boldsymbol{w}) = -\sum_{i=1}^{M}\sum_{j=1}^{N} l_j^{(i)} \log \hat{l}_j^{(i)}, \tag{7}$$

where $\boldsymbol{w}$ represents the model parameters. The model parameters are trained with multiple epochs. In each epoch, a stochastic gradient descent optimizer is utilized, which calculates the gradient $\nabla f(\boldsymbol{w})$ of a mini-batch in each iteration and updates the weights as

$$\boldsymbol{w} = \boldsymbol{w} - \eta \cdot \nabla f(\boldsymbol{w}), \tag{8}$$

where η is a pre-determined learning rate.

III. Neural Architecture Design with NAS

In this section, we introduce the differentiable architecture search method to automatically design a deep neural network for the constructed dataset [13]. Specifically, the method consists of three components, i.e., architecture space, evaluation method, and search strategy. The architecture space defines the set of allowed architectures. The evaluation method refers to the process of estimating the performance of a specific architecture. The search strategy details the algorithm that explores the architecture space. The implementation details are explained as follows.

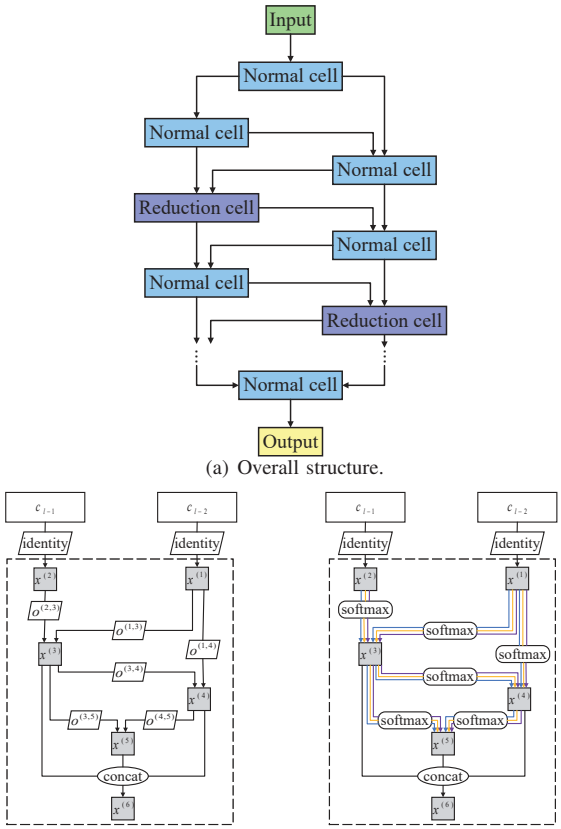
A. Architecture Space

As shown in Fig. 2(a), the overall structure of the neural network is manually predetermined, where normal and reduction cells are alternatively stacked in series. As shown in Fig. 2(b), the architecture space is defined for each type of cells by using a directed acyclic graph, where each node $x^{(i)}$ is a latent representation and each edge (i, j) is associated with an operation $o^{(i,j)}(x)$ that transforms $x^{(i)}$. We assume that the number of nodes is uniformly given by K for all cells.

Note that the directed acyclic graph defines a partial order relation between nodes. We can order the nodes in a unique sequence $x^{(1)}, x_l^{(2)}, \ldots, x^{(K)}$, where each node is determined by its predecessors, i.e.,

$$x^{(i)} = \sum_{i<j} o^{(i,j)}\left(x^{(j)}\right). \tag{9}$$

Thus, the architecture space is defined by the combination of the operations associated with each edge. For each cell in layer l, we assume there are two input nodes $x^{(1)}$ and $x^{(2)}$ that directly carry the outputs of cells c_{l-2} and c_{l-1}, respectively, and a single output node $x^{(K)}$ that applies a reduction operation to all the intermediate nodes.



(a) Overall structure.

(b) Cell structure with discrete architecture space.

(c) Cell structure with continuous architecture space.

Fig. 2. An overview of the deep neural network.

To reduce the computational complexity, we relax the discrete architecture space to a continuous one as in Fig. 2(c), where the particular operation associated with each edge is replaced by a softmax over all possible operations, i.e.,

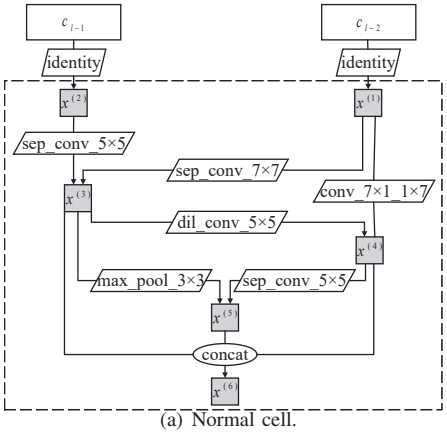
$$\bar{o}^{(i,j)}(x) = \sum_{o\in\mathcal{O}} \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o'\in\mathcal{O}} \exp\left(\alpha_{o'}^{(i,j)}\right)} o(x). \tag{10}$$

where $\mathcal{O}$ is a set of candidate operations. Note that operation $\bar{o}^{(i,j)}(x)$ can be represented by a vector $\boldsymbol{\alpha}^{(i,j)}$ of dimension $|\mathcal{O}|$. The architecture search reduces to finding a set of continuous variables $\mathcal{A} = \{\boldsymbol{\alpha}^{(i,j)}\}_{(i,j)}$. At the end, a discrete architecture can be obtained by rounding the continuous operation to the most likely one, i.e.,

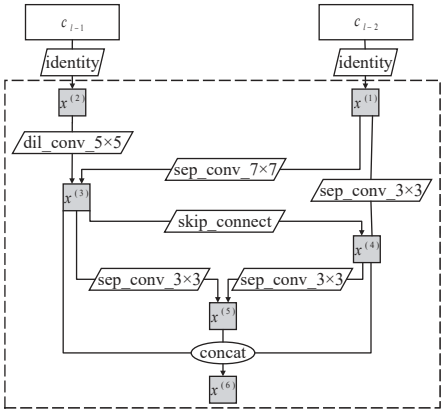
$$o^{(i,j)}(x) = \arg\max_{o\in\mathcal{O}} \alpha_o^{(i,j)}. \tag{11}$$

B. Evaluation Method

For a particular architecture $\mathcal{A}$, it can be evaluated by the loss $f_{\text{val}}(\boldsymbol{w}^*(\mathcal{A}), \mathcal{A})$ on the validation set, where $\boldsymbol{w}^*(\mathcal{A}) = \arg\min_{\boldsymbol{w}} f_{\text{train}}(\boldsymbol{w}, \mathcal{A})$ is the optimal model parameters on the training set. However, the training process can be expensive, which highly prohibits the exploration of the architecture space. Thus, to avoid the entire training process, we replace



(a) Normal cell.



(b) Reduction cell.

Fig. 3. Discovered cell structures with neural architecture search.

the optimal model parameters $\boldsymbol{w}^*(\mathcal{A})$ with the one-step parameters, given by

$$\boldsymbol{w}' = \boldsymbol{w} - \eta \cdot \nabla_{\boldsymbol{w}} f_{\text{train}}(\boldsymbol{w}, \mathcal{A}), \tag{12}$$

where $\boldsymbol{w}$ is the current model parameters maintained by the search strategy. The architecture $\mathcal{A}$ is then evaluated by using

$$v(\mathcal{A}) = f_{\text{val}}(\boldsymbol{w}', \mathcal{A}). \tag{13}$$

C. Search Strategy

To minimize $v(\mathcal{A})$, we evaluate the architecture gradient by applying chain rule to (13) and yield

$$\begin{aligned}\nabla v(\mathcal{A}) =& \nabla_{\mathcal{A}} f_{\text{val}}(\boldsymbol{w}', \mathcal{A}) \\ &- \eta \cdot \nabla^2_{\mathcal{A},\boldsymbol{w}} f_{\text{train}}(\boldsymbol{w}, \mathcal{A}) \cdot \nabla_{\boldsymbol{w}'} f_{\text{val}}(\boldsymbol{w}', \mathcal{A}).\end{aligned} \tag{14}$$

To further reduce the computational complexity, we use finite difference approximation for the second term. Let ϵ be a small scalar and set

$$\boldsymbol{w}^{\pm} = \boldsymbol{w} \pm \epsilon \cdot f_{\text{val}}(\boldsymbol{w}', \mathcal{A}). \tag{15}$$

We have

$$\begin{aligned}&\nabla^2_{\mathcal{A},\boldsymbol{w}} f_{\text{train}}(\boldsymbol{w}, \mathcal{A}) \cdot \nabla_{\boldsymbol{w}'} f_{\text{val}}(\boldsymbol{w}', \mathcal{A}) \\ &\quad\approx \frac{\nabla_{\mathcal{A}} f_{\text{train}}(\boldsymbol{w}^+, \mathcal{A}) - \nabla_{\mathcal{A}} f_{\text{train}}(\boldsymbol{w}^-, \mathcal{A})}{2\epsilon}.\end{aligned} \tag{16}$$

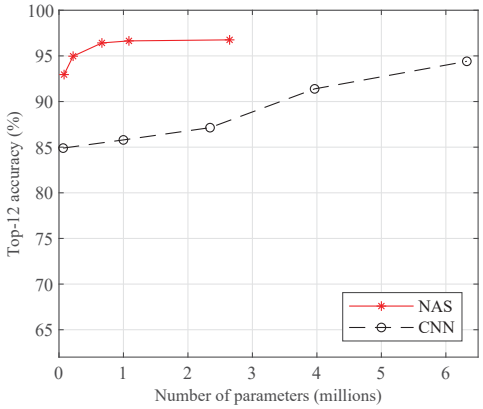


Fig. 4. Model accuracy of both the discovered network and the plain convolutional neural network.

Thus, we can update the architecture as

$$\mathcal{A} = \mathcal{A} - \xi \cdot \nabla v(\mathcal{A}), \tag{17}$$

where ξ is a pre-determined learning rate.

IV. Numerical Results

In this section, we compare the NAS-based model with other hand-crafted models [16]. The number of nodes in each cell is uniformly set as $K = 6$. Specifically, we consider 9 classical neural operations, including identity, 3×3 average pooling, 3×3 max pooling, 3×3 dilated convolution, 5×5 dilated convolution, 3×3 separable convolution, 5×5 separable convolution, 7×7 separable convolution, 1×7 then 7×1 convolution, and a zero operation indicating the associated edge is not connected. The model accuracy is represented by the top-12 accuracy, which represents the probability that the optimal antenna configuration is within the top 12 highest probability configurations given by the model output. The computational complexity is represented by the number of parameters and the memory usage.

In Fig. 3, we show the discovered cell architectures by using NAS. Specifically, 6 different operations and 7 connections are applied in both the normal and reduction cells. As compared to hand-crafted architectures, the discovered cells exhibit more sophisticated architectures with more types of operations and more connections between the representation nodes.

In Fig. 4, we show the top-12 accuracy of the discovered architecture and the plain convolutional neural network as a function of the number of parameters. As we can see, the model accuracy increases with the number of parameters, as the representative capability is enhanced by the model complexity. As compared to the convolutional neural network, the discovered architecture achieves higher accuracy with the same amount of parameters, which implies it has higher parameter efficiency. In addition, the curve of the discovered architecture converges quickly at the point with 1 M parameters, indicating that NAS can learn a relatively small neural network with high accuracy for the power coverage optimization problem.

TABLE I
COMPARISON WITH HAND-CRAFTED MODELS.

Architecture	Parameters (M)	Memory (MB)	Top-12 Accuracy (%)
All-CNN	1.4	38.7	82.66
SqueezeNet	0.8	108.2	89.04
NiN	2.2	7.5	90.94
RiR	9.8	-	92.28
MobileNet	3.3	123.5	93.18
GoogleNet	6.4	302.4	94.30
ShuffleNet	1.0	327.8	94.30
ShuffleNetv2	1.3	149.3	94.85
MobileNetv2	2.3	201.1	96.08
DenseNet	12.6	1309.1	96.41
Wideresidual	55.9	1346.1	96.86
NAS	0.2	46.3	94.97
	1.1	244.5	96.65
	8.5	552.3	97.32

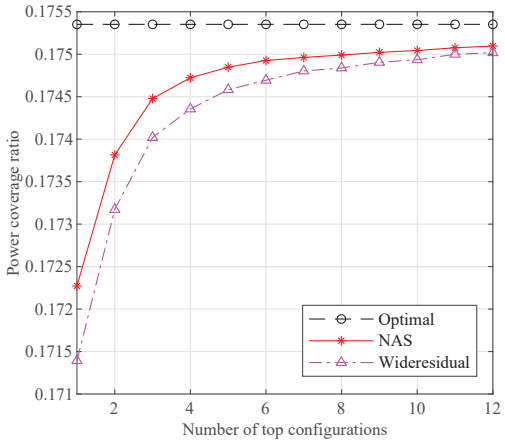


Fig. 5. Power coverage ratio as a function of the number of top configurations.

In Table I, we compare the discovered architecture with other classical hand-crafted neural networks [16], in terms of the parameter number, the memory usage and the top-12 accuracy. As we can see, the discovered architecture outperforms most hand-crafted architectures with only 0.2 M parameters and 46.3 MB memory, except for the MobileNetv2, DenseNet and Wideresidual. As we slightly increase the model scale to 1.1 M parameters and 244.5 MB memory, the discovered architecture outperforms the MobileNetv2 and DenseNet with lower computational complexity. As we further increase the mode scale to 8.5 M parameters and 552.3 MB memory, the discovered architecture outperforms the Wideresidual and still maintains much lower computational complexity. Thus, the discovered architecture shows structural advantages as compared to hand-crafted models.

In Fig. 5, we show the highest power coverage ratio among the top configurations as a function of the number of top configurations. As we examine more configurations suggested by the neural network, the highest coverage ratio increases monotonically. The discovered network outperforms the best hand-crafted model, i.e., the Wideresidual network, and achieves a near-optimal performance by examining the top 6 highest probability configurations.

V. CONCLUSION

In this article, we considered the power coverage optimization of a single small cell in urban environments. We propose a NAS-based method to automatically design a deep neural network that directly outputs the suggested antenna configuration in azimuth and downtilt by using the map information. Specifically, we use a commercial radio propagation software with real city maps to construct our dataset, and adopt a differential architecture search method to design a problem-dependent neural architecture. Numerical results show that the discovered architecture involves much more types of operations and more connections as compared to hand-crafted architectures, and shows superior performance in terms of model accuracy, parameter efficiency, memory usage and the actual power coverage ratio.

REFERENCES

[1] S. Wang and R. Chen, "Rethinking cellular network planning and optimization," *IEEE Wireless Commun.*, vol. 23, no. 2, pp. 118–125, Apr. 2016.

[2] W. Zhao *et al.*, "Approximation algorithms for cell planning in heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 1561–1572, Feb. 2017.

[3] L. Shen and S. Wang, "Monte Carlo tree search for network planning for next generation mobile communication networks," in *Proc. GLOBECOM'21*, Madrid, Spain, Dec. 2021.

[4] I. Siomina, P. Varbrand, and D. Yuan, "Automated optimization of service coverage and base station antenna configuration in UMTS networks," *IEEE Wireless Commun.*, vol. 13, no. 6, pp. 16–25, Dec. 2006.

[5] S. Hurley *et al.*, "Modelling and planning fixed wireless networks," *Wireless Netw.*, vol. 16, no. 3, pp. 577–592, Apr. 2010.

[6] N. Lakshminarasimman *et al.*, "Evolutionary multiobjective optimization of cellular base station locations using modified NSGA-II," *Wireless Netw.*, vol. 17, no. 3, pp. 597–609, Apr. 2011.

[7] R. Han *et al.*, "Coverage optimization for dense deployment small cell based on ant colony algorithm," in *Proc. IEEE VTC'14-Fall*, Vancouver, BC, Canada, Sep. 2014.

[8] C. Shen *et al.*, "Generalized global bandit and its application in cellular coverage optimization," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 218–232, Feb. 2018.

[9] Y. Liu *et al.*, "An efficient stochastic gradient descent algorithm to maximize the coverage of cellular networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 7, pp. 3424–3436, Jul. 2019.

[10] E. Balevi and J. G. Andrews, "Online antenna tuning in heterogeneous cellular networks with deep reinforcement learning," *IEEE Trans. Cogn. Netw.*, vol. 5, no. 4, pp. 1113–1124, Dec. 2019.

[11] M. E. Morocho-Cayamcela, M. Maier, and W. Lim, "Breaking wireless propagation environmental uncertainty with deep learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 8, pp. 5075–5087, Aug. 2020.

[12] J. Fang *et al.*, "FNA++: Fast network adaptation via parameter remapping and architecture search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 9, pp. 2990–3004, Sep. 2020.

[13] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *Proc. ICLR'19*, New Orleans, LA, USA, May 2019.

[14] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 1, pp. 1997–2017, Mar. 2019.

[15] R. Hoppe, G. Wölfle, and U. Jakobus, "Wave propagation and radio network planning software WinProp added to the electromagnetic solver package FEKO," in *Proc. ACES'17*, Florence, Italy, Mar. 2017.

[16] D. Su *et al.*, "Is robustness the cost of accuracy? – a comprehensive study on the robustness of 18 deep image classification models," in *Proc. ECCV'18*, Munich, Germany, Sep. 2018.