A Refined Energy Optimization Model for Edge Computing with Machine Learning

Xian Liu
Department of Systems Engineering
University of Arkansas at Little Rock, USA

Changcheng Huang
Department of Systems and Computer Engineering
Carleton University, Ottawa, Canada

Wilsun Xu
Department of Electrical and Computer Engineering
University of Alberta, Edmonton, Canada

***Abstract* – Edge computing (EC) with artificial intelligence (AI) / machine learning (ML) is a promising paradigm in current 5G-Advanced and future 6G wireless technologies. Energy optimization is a primary issue in most EC/ML systems. In this paper, a refined model is developed to seek optimal energy allocation. The present work introduces several features which were often omitted in existing studies, including fine-grained discrete optimization, non-singular CPU cycle allocation, long-tailed data traffic, and non-singular pathloss terms. The energy optimization model is solved by the ML methodology and compared with a conventional solver. Simulations showed that the efficiency may be improved more than 90%.**

***Index Terms* – Edge intelligence, machine learning, mobile edge learning, offloading.**

I. Introduction

Multi-access edge computing (a.k.a. *mobile edge computing*) (MEC) has gained much attention as the 4G technology evolved to 5G [1-2]. An MEC system usually includes multiple *edge devices* (EDs). Commonly, these EDs are limited by scarce computational and storage capacities. In order to meet the primary QoS criteria (e.g., latency upper bound), an ED often needs to offload some computing-intensive jobs to *edge servers* (ES). For a given configuration, it is always important to seek some optimal offloading schemes. Due to the real-time and mobile features of most MEC paradigms, it is also highly desirable to find some lightweight optimization schemes. Therefore, two aspects, effective optimization models and efficient solving strategies, should be concurrently investigated.

Edge intelligence (EI) represents one of most promising initiatives in the *beyond-fifth-generation* (B5G) wireless communication technologies [3-4]. In nature, EI is an interdisciplinary initiative that integrates the state-of-the-art of *artificial intelligence* (AI) into MEC. Currently, there are several important areas in EI worth further explorations, e.g., *resource optimization with machine learning* (ROML).

In a mathematical viewpoint, ROML can be regarded as an evolved version of the *wait-and-see* (WS) model in stochastic programming [5]. In WS, some parameters are the samples of realized random variables. For each scenario, there is a corresponding optimal solution in terms of decision variables such as offloading portion, CPU cycles, and transmission power. The optimization solving process in WS can be computational expensive due heterogeneity of the sample space. With a proper architecture, ROML can significantly improve the computational efficiency by emulating the optimization procedure, while retaining a satisfactory accuracy.

In the recent literature, a great deal of attention has been paid to the formulating MEC and developing heuristic optimization procedures. However, some important issues were not well addressed in most existing works. The purpose of the present study is twofold: developing an optimization model to include some often-omitted issues and seeking possibly lightweight solving schemes. One of the main issues in modeling is to consider the *partial offloading* (PO) mode with fine-grained weights. However, as a practical scheme, these weights should be characterized by discrete variables. Besides, the long-tailed feature of data traffic should not be neglected. These issues will be well included into this study.

The rest of this paper is organized as follows. The system model is introduced in Section II, along with a notation list. Then, in Section III, some relevant works are reviewed and the main features of the present study are summarized. Next, the computation modes in MEC are described in Section IV. Then, an optimization model is formulated in Section V, with several remarks for its deeper analytical attributes. Sections VI and VII are devoted for machine learning, where several standard plots are provided and the computational performance is compared. Finally, the paper is concluded in Section VIII.

II. System Model

In the present study, we consider a small cell in which there are N edge devices (EDs) and one processing hub (Fig. 1).

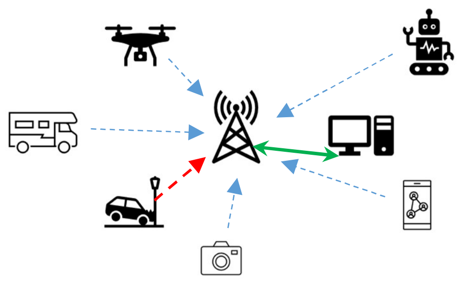

Figure 1. Generic view of MEC system.

Here the processing hub is an abstraction of a variety of communication nodes, e.g., macro base station, micro base station, *remote radio head* (RRH), *roadside unit* (RSU), etc. However, we assume that such a processing hub is equipped with or connected through a high-bandwidth physical link to a dedicated edge server (ES).

TABLE I. LIST OF MAIN NOTATIONS

b_{ij}	Level of partial offloading
d_0	Reference distance of d_i
d_i	Distance between ED and ES (m)
f_i^l	ED resource for job i (CPU cycles per second)
h_i	Channel gain of small scale fading
m_i	Nakagami-m factor
x_i	Discrete offloading weight
β	Path loss exponent
C_i	Total number of CPU cycles of job i
D_i	Data size of job i (bits)
F_i	ES resource for job i (CPU cycles per second)
M_i	Offloading cardinality of job i
N_0	Power spectral density (Watt/Hz)
P_i	Transmission power of ED i (Watt)
P_i^w	Idle power of ED i (Watt)
R_c	Cell radius (m)
R_i	Transmission rate (bits/second)
W_c	Bandwidth in the cell (Hz)
W_i	Bandwidth per channel (Hz)

In this system, the index of EDs is denoted as $i \in \mathbb{N} = \{1, 2, ..., N\}$. For the ED $i \in \mathbb{N}$, the associated computational job is characterized by a triplet $T_i \overset{def.}{=} \{D_i, C_i, \tau_{i,ub}\}$, where D_i is the data size (bits), C_i is the total number of required CPU cycles, and $\tau_{i,ub}$ is the upper bound of latency. Note that this triplet or equivalent patterns represent the metrics commonly adopted in the literature of MEC (ref.). It is an easy notion that D_i is a *random variable* (RV). On the other side, the CPU cycles per instruction depend on the type of instructions, while a computational procedure consists of various types of instructions. Accordingly, it is also reasonable to consider C_i as an RV. The upper bound of latency $\tau_{i,ub}$ could be an RV or a fixed threshold depending on the concerned application. Note that the ratio C_i / D_i characterizes the *computation intensity* (CPU cycles per bit).

In the MEC problems involving offloading, one of the primary decision variables is the *offloading weight*. For the system consisting of N EDs, the offloading weight is a vector of N components. In principle, the simplest scheme is *binary offloading* (BO). In BO, a job is either completely done in an ED or completely done in an ES. A generalization of BO is referred to as partial offloading (PO). If PO is fine-grained, then the job i can be offloaded with one of different thresholds, denoted as b_{ij} $(j = 1, 2, ..., M_i)$. These thresholds could be estimated by checking the internal segments of the concerned software task, e.g., code or data. Note that these thresholds comprises a discrete list, but the item b_{ij} may take a real value in [0,1), e.g., $(b_{i1}, b_{i2}, b_{i3}) = (0.12,\ 0.57,\ 0.95)$. This is the scheme on which the present study is based. Accordingly, the offloading weight is defined as a vector and denoted as $x = (x_1, x_2, ..., x_N)$, whose components $x_i \in \{b_{i1}, b_{i2}, ..., b_{iM_i}\}$.

III. RELATED WORKS

A great deal of studies on MEC offloading has been reported in the literature. A comprehensive review can be found in [6]. In a high level, the offloading applications may be practiced in either downlink or uplink. One of the well-known problems in downlink offloading is the cache replacement paradigm (see [7] and the references therein). On the other hand, as described in the preceding section, the paradigm investigated in this paper is one of the uplink offloading problems. In particular, the present study focuses on the issue of energy optimization subject to some latency thresholds. Several relevant works were recently reported. In [8], the energy consumption problem was formulated as a *mixed integer nonlinear programming* (MINLP) problem, where three sets of decision variables were considered: offloading weights, ES resources, and wireless channels. Specifically, their offloading weights were binary $\{0,1\}$. In [9], a simplified formulation was described, where the channel allocation was fixed and the offloading weights were continuous in [0, 1). However, the machine learning feature was included in [9] while not in [8]. More studies in this area can be found in the reference lists of [8] and [9]. In general, however, it seems that several important issues have not been sufficiently investigated. Some concerns are itemized as follows:

- The random samples of primary parameters, e.g., the data size, were generated from over-simplified distributions, such as the uniform distribution or Gaussian distribution.
- The offloading weights were coarse-grained: either binary or continuous over the unit interval.
- The singularity pitfall of CPU cycle allocation was not explicitly avoided in the base models.
- The statistics of small-scale and large-scale channel fading were not well addressed.
- Only a small portion of studies involved machine learning.

The present work tries to pay more attentions to aforementioned issues.

- The random samples of the data size were generated from long-tailed distributions popularly found in Internet ([7], [10-12]).

- The offloading weights were fine-grained, as described in the preceding section.
- The singularity issue of CPU cycle allocation was clearly avoided in the base model.
- For the small-scale fading, instead of the basic *Rayleigh* model (i.e., the fading power following exponential distribution), the more general *Nakagami-m* model is adopted. Moreover, for the path-loss fading, the singularity is avoided and the path loss exponent can be arbitrary, rather than restricted to integers.
- Finally, a machine learning approach is developed.

IV. Computational Modes

In the following, we discusses two typical modes in MEC. Both energy consumption and processing latency are relevant.

A. Local Processing

Let f_i^l be the CPU cycles per second in the ED *i*. A commonly adopted formula for evaluate f_i^l is based on the *dynamic voltage scaling* (DVS) approach [13]:

$$f_i^l = \min\left\{\frac{(1-x_i)C_i}{\tau_{i,ub}}, f_{i,\max}\right\}, \tag{1}$$

where $f_{i,\max}$ is the maximum CPU frequency of ED *i*. The energy consumed by CPU is directly proportional to the squared-voltage for the chip, V^2, while V is linearly associated with f_i^l. Accordingly, The energy per CPU cycle can be expressed as $\kappa(f_i^l)^2$, where κ is the chip-dependent parameter (e.g., effective switched capacitance). A typical value is $\kappa \approx 10^{-26}$. Therefore, the total energy consumed by C_i cycles with the effect of x_i is:

$$E_i^l = \kappa(f_i^l)^2(1-x_i)C_i. \tag{2}$$

On the other hand, the incurred latency is:

$$t_i^l = (1-x_i)C_i / f_i^l. \tag{3}$$

B. Remote Processing

Remote processing is carried out in edge servers. One of the components of latency is due to uplink transmissions. The transmission rate takes the following form:

$$R_i = W_i \log_2\left(1+\frac{h_i P_i}{N_0 W_i (d_0+d_i)^\beta}\right). \tag{4}$$

Accordingly, the transmission time is:

$$t_i^{tr} = x_i D_i / R_i. \tag{5}$$

On the other hand, the CPU processing time is:

$$t_i^s = x_i C_i / F_i. \tag{6}$$

The consumed energy of RF for transmission is

$$E_i^{tr} = P_i t_i^{tr} = x_i D_i P_i / R_i. \tag{7}$$

Moreover, the energy of ED while waiting for ES is:

$$E_i^w = P_i^w t_i^s = x_i C_i P_i^w / F_i. \tag{8}$$

V. Problem Formulation

In this paper, our interest is to minimize the energy consumption with multiple constraints. The optimization problem is formulated as follows.

(OFLD_OPTM1)

$$\underset{x,F}{\text{minimize}} \quad z = \sum_{i=1}^{N} (E_i^l + E_i^{tr} + E_i^w); \tag{9}$$

s.t.

$$t_i^l = (1-x_i)C_i / f_i^l \le \tau_{i,ub}; \tag{10}$$

$$t_i^{tr} + t_i^s = \frac{x_i D_i}{R_i} + \frac{x_i C_i}{F_i} \le \tau_{i,ub}; \tag{11}$$

$$0 < \frac{F_{\min}}{F_{\max}} \le \frac{F_i}{F_{\max}} \le 1; \tag{12}$$

$$\sum_{i=1}^{N} \frac{F_i}{F_{\max}} \le 1; \tag{13}$$

$$x_i \in \{b_{i1}, b_{i2}, ..., b_{iM_i}\}. \qquad (i = 1,2,...,N) \tag{14}$$

Several remarks are instructive to gain deeper insights.

Remark 1: The decision variables of NEC_OPT1 are continuous F_i and discrete x_i $(i = 1,2,...,N)$.

Remark 2: OFLD_OPTM1 is a nonlinear, non-convex, and hybrid optimization problem.

Remark 3: The discrete constraint (14) can be dealt with the conventional combinatorial strategies, usually provided in a standard integer nonlinear programming solver. However, if such a standard solver is not available or not scalable, it is desirable to use the common nonlinear programming solvers provided in most engineering software products, such as Matlab. In this situation, a method proposed in [14] or [15] can be customized.

Remark 4: Due to its positive limit on the left-hand side, constraint (12) is not redundant for (13).

Remark 5: The lower bound of constraint (12) avoids the unbounded pitfall of (6), (8), and (11).

Remark 6: OFLD_OPTM1 is a quasi-separable programming problem. In the context of nonlinear optimization, the definition of the separable programming problem also includes the constraints and convexity [16, Ch. 13]. In OFLD_OPTM1, due to eqs. (2), (7), and (8), the objective function is separable. Note that, it is possible $b_{i1} \ne 0$, since some software tasks may include a non-offloadable portion. For example, a task for face recognition may include *K* Java classes, of which *K-1* are offloadable.

It is noted that Remarks 3 through 6 have not well been addressed in the open literature.

VI. Simulation Setup

A. Distribution of Edge Devices

As shown in Fig. 1, we consider a network with N EDs and one processing hub. The edge sever is associated with this hub. These N EDs are independently uniformly distributed in the disc with radius R_c. Accordingly, the *probability density function* (PDF) is $1/(\pi R_c^2)$. In the polar coordinate system, the *cumulative distribution function* (CDF) is:

$$P_R(R \le r, \Theta \le \theta) = \int_0^{\theta}\int_0^{r} \frac{1}{\pi R_c^2}\rho d\rho d\phi = \frac{\theta r^2}{2\pi R_c^2}. \qquad (15)$$

It is worth noting that the above model is equivalent to the homogeneous *Poisson point process* (PPP) conditioned that there are N points within a finite disc [17, Theorem 2.9].

B. Distribution of Small-Scale Fading

For the small-scale fading, we assume that its channel gains follow the squared Nagagami-m distribution with unit mean. The random data are generated from the *inverse incomplete gamma* function.

C. Distribution of Data Size

The data size D_i is assumed to follow the Pareto distribution ([7], [10]). The CDF of Pareto distribution is as follows:

$$F_D(s) = 1 - \left(h_p / s\right)^{c_p}. \quad (s \ge h_p > 0;\ c_p > 0) \qquad (16)$$

As indicated in [10], various data communication entities can be well described by (16) with $1 < c_p < 2$. Note that (16) is one of the refinements against most existing works.

VII. Neural Network Configuration

One of the most popular realizations for the ANN notion is the *feed-forward neural network* (FNN) [19, Ch. 6]. In the present work, we implement an FNN for the MEC problem. One of the axioms of FNN is to start trying small architectures whenever possible. We have investigated several small architectures. In this study, the FNN consist of three hidden layers with 32, 16, and 16 neurons, respectively. The total number of input vectors is chosen to be larger than total weights of this FNN. The rationale is well described in [19]. The datasets are generated by a standard nonlinear optimization solver in Matlab. In particular, the target dataset is composed of those converged solutions of OFLD_OPTM1.

A. Parameters of Edge Computing

The main parameters are listed in Tables II, where the index $i = 1, 2, \ldots, N$, and $U(a,b)$ stands for the uniform distribution in $[a,b]$.

TABLE II. Main Parameters in Testbed

Parameter	Value
$\{b_{i1}, b_{i2}, b_{i3}, b_{i4}\}$	$\{0, 0.35, 0.70, 0.95\}$
c_p	1.67
h_p	2×10^5 (bits)
$f_{\max}^{l}$	2 GHz
m_i; β	1.2; 3.5
$F_{\max}$	30 GHz
C_i	$U(900,1100)$ (Mega cycles)
M_i; N	4; 8
N_0	-143.97 (dBm/Hz)
P_i; P_i^w	480 (mW); 100 (mW)
R_c	100 (m)
W_c	20 (MHz)

In the configuration of FNN, 80%, 15%, and 5% of samples are used as the datasets for training, validation, and internal testing, respectively. In the training stage, the *Levenberg-Marquardt* (LM) algorithm was adopted as the training method. In the context of FNN, the LM algorithm is usually the first choice [18, 19]. For the hidden layers, two of the most popular activation functions are used: the *log-sigmoid* function and the *rectified linear unit* (ReLU) function. For the output layer, the *pure linear* function is used.

A. Training

The profiles of training are illustrated in Fig. 2 through Fig. 4. Several interesting yet important insights can be gained from above results. The performance of *mean square error* (MSE) of training, validation, and internal testing are illustrated in Fig. 2. Next, the corresponding error profile is presented in Fig. 3, where the ordinate represents the number of errors that falls within each interval on the abscissa. It is observed that most instances fall in two very narrow error intervals. As a result, this NN has done a successful training.

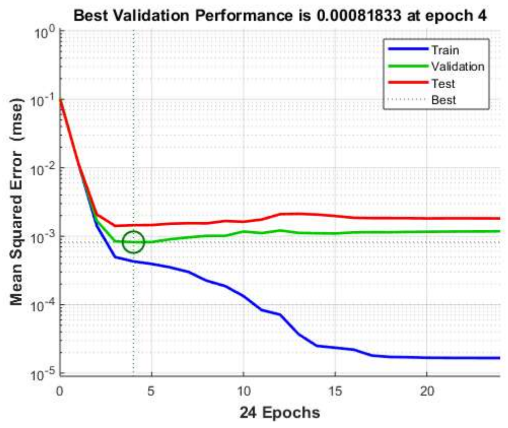

Figure 2. MSE performance.

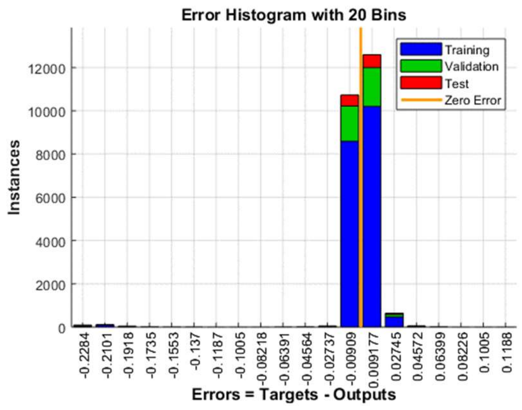

Figure 3. Histogram of errors.

B. *External Testing*

In total, 1530 samples are used for FNN (32, 16, 16). The CDF profile of optimal solutions produced by the trained FNN are illustrated in Fig. 4, where the top plot is the CDF of the mean of optimal x_i, while the bottom plot is the CDF of the mean of optimal $F_i / F_{\max}$. As a comparison, the corresponding CDF profiles generated by a standard *nonlinear programming* (NLP) solver are illustrated in Fig. 5. This NLP solver has been upgraded to deal with both discrete and continuous constraints, as shown in eqs. (10) through (14). Note that the trained FNN is well validated by the range-match between Figs. 4 and 5 (except the illustrations of curve shapes are affected by the different horizontal scales/spans).

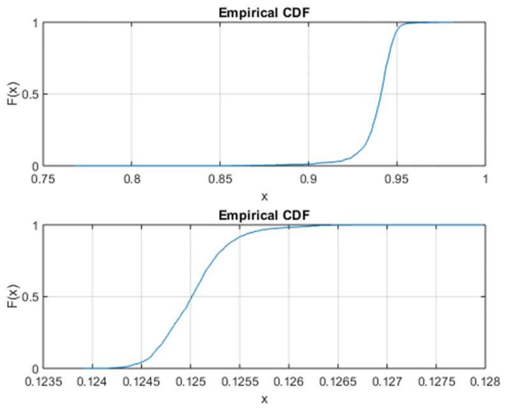

Figure 4. CDF of optimal solutions from trained FNN.

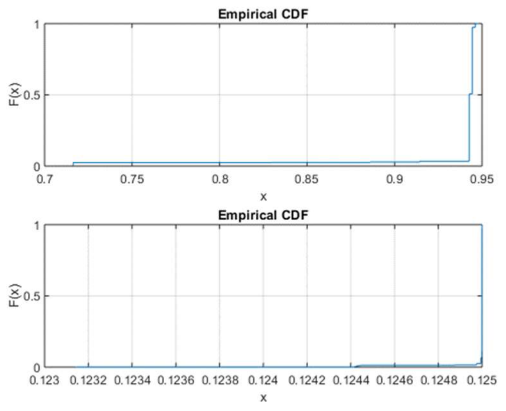

Figure 5. CDF of optimal solutions from NLP solver.

C. *Computational Efficiency*

In this study, extensive numerical experiments have been carried out by the trained neural network. All training and testing procedures were conducted in a computer equipped with the 3.50 GHz quad-core CPU and 16 GB RAM. Three sets of representative results are presented In Table III. The individual datum is the average over is the entire 1530 samples. It is observed that the computational time of FNN has reduced to 7% or 8%.

TABLE III. AVERAGE COMPUTATIONAL TIME

NLP sol. (ms)	88.5521	81.8689	83.1001
FNN test (ms)	7.1221	5.9344	5.8141
Ratio (%)	8.0428	7.2487	6.9965

VIII. CONCLUSION

It is imperative to refine the resource optimization schemes for MEC offloading paradigms. The purpose of this paper is twofold: developing an optimization model to include some often-omitted issues and seeking some lightweight solving schemes. There has been an interest in investigating the effectiveness of a small-scale FNN for the second purpose. The present study shows that a small FNN is functional quite well. In the ongoing project, the main attention will be paid to training FNN to solve large-scale MEC problems.

REFERENCES

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A survey on mobile edge computing: the communication perspective,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 4th Quarter 2017.

[2] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, “A survey on the edge computing for the internet of things,” *IEEE Access*, vol. 6, pp. 6900-6919, November 2017.

[3] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, “Toward an intelligent edge: wireless communication meets machine learning,” *IEEE Communications Magazine*, vol. 58, no. 1, pp. 19-25, January 2020.

[4] Y. Xiao, G. Shi, Y. Li, W. Saad, and H. V. Poor, “Toward self-learning edge intelligence in 6G,” *IEEE Communications Magazine*, vol. 58, no. 12, pp. 34-40, December 2020.

[5] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming* (2nd ed). New York: Springer, 2011.

[6] B. Wang, C. Wang, W. Huang, Y. Song, and X. Qin, “A survey and taxonomy on task offloading for edge-cloud computing,” *IEEE Access*, vol. 8, pp. 186080-186101, October 2020.

[7] C. Zhong, M. C. Gursoy, and S. Velipasalar, “Deep multi-agent reinforcement learning based cooperative edge caching in wireless networks,” in *Proc. of IEEE International Conference on Communications* (ICC), 2019.

[8] P. Zhao, H. Tian, C. Qin, and G. Nie, “Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing,” *IEEE Access*, vol. 5, pp. 11255-11268, June 2017.

[9] J. Li and T. Lv, “Deep neural network based computational resource allocation for mobile edge computing,” in *Proc. of IEEE Globecom Workshops*, 2018.

[10] K. Park and W. Willinger (eds.), *Self-Similar Network Traffic and Performance Evaluation*, Wiley, NY, 2000.

[11] Y. Geng, W. Hu, Y. Yang, W. Gao, and G. Cao, “Energy-efficient computation offloading in cellular networks,” in *Proc. of IEEE 23rd International Conference on Network Protocols* (ICNP), 2015.

[12] H. Volos, T. Bando, and K. Konishi, “Latency modeling for mobile edge computing using LTE measurements,” in *Proc.* of *IEEE 88th Vehicular Technology Conference* (VTC-Fall), 2018.

[13] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, “Mobile-edge computing: partial computation offloading using dynamic voltage scaling,” *IEEE Trans. Commun*., vol. 64, no. 10, pp. 42684282, Oct. 2016.

[14] X. Liu, “On compact formulation of constraints induced by disjoint prohibited-zones”, *IEEE Transactions on Power Systems*, vol. 25, no. 4, pp. 2004-2005, 2010.

[15] X. Liu, “A new approach for economic dispatch with disjoint feasible zones: model and solution”, in *Proc. of IEEE Power and Energy Society General Meeting* (PESGM), 2016.

[16] F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research* (10th ed.), McGraw-Hill Higher Education, 2014.

[17] M. Haenggi, *Stochastic Geometry for Wireless Networks*, Cambridge University Press, 2012

[18] Matlab, *Neural Network Toolbox*, R2019a.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.